PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/60

A2

(11) International Publication Number: **WO 00/42544**

(43) International Publication Date: 20 July 2000 (20.07.00)

(21) International Application Number: PCT/US00/01084

(22) International Filing Date: 18 January 2000 (18.01.00)

(30) Priority Data:
09/232,357 15 January 1999 (15.01.99) US

(71) Applicant: IMANDI CORPORATION [US/US]; 14570 NE 95th Street, Redmond, WA 98052 (US).

(72) Inventors: JOHNSON, Eric, W., W.; 16911 NE 106th Street, Redmond, WA 98052 (US). KHER, Raghav, P.; 17436 NE 38th Street, Redmond, WA 98052 (US). JACOBS, Bradley, W.; 29824 – 25th Place South, Federal Way, WA 98003 (US).

(74) Agent: BERGSTROM, Robert, W.; Weiss Jensen Ellis & Howard, Suite 2600, 520 Pike Street, Seattle, WA 98101 (US).

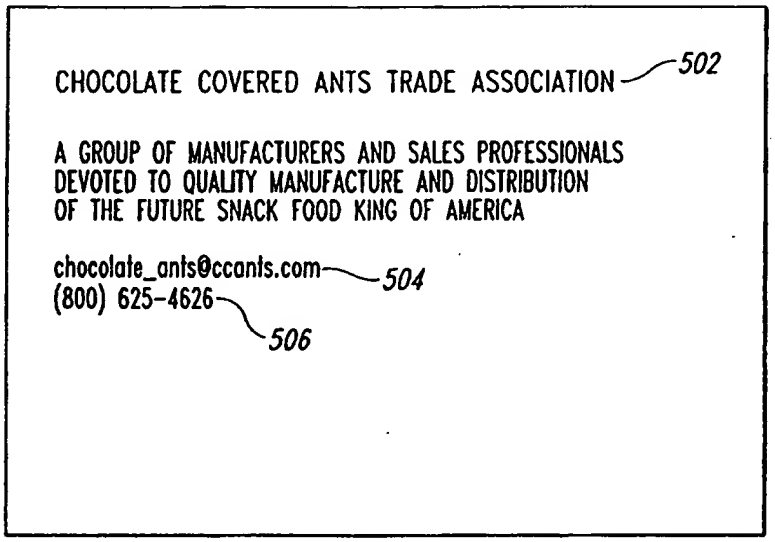
(81) Designated States: AU, BR, CA, CN, IN, JP, KR, NO, NZ, SG, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: EXTRACTION OF VENDOR INFORMATION FROM WEB SITES

(57) Abstract

A database and database creation, maintenance, and update processes and tools for storing vendor information for use in technology–enabled markets. The vendor information stored within the database allows for automated compilations of lists of vendors having an arbitrary geographical proximity to a customer, offering a product or service desired by the customer, and meeting various customer preferences. Database creation and update tools extract information from various information sources, such as Internet–based web sites, and enhance and update the database on a continuous basis.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

Extraction of Vendor Information from Web Sites

Technical Field

The present invention relates to technology-enabled retail and wholesale markets, and, in particular, to a method and system for extracting information about merchants from Internet-based web sites and organizing the extracted information to facilitate subsequent searching and retrieval operations needed by technology-enabled retail and wholesale markets.

Background of the Invention

Computer-readable compilations of vendor information are important components of new, Internet-based retail and wholesale markets. Such compilations may be stored as text files or collections of hyper-linked web pages, but are most commonly stored in any of a number of different types of computer database management systems, such as relational database management systems and object-oriented database management systems. Printable and human-readable representations of the information contained in the compilations can be generated automatically by any of a number of different database management system recording tools. The information contained in many vendor databases is also accessible via graphical displays generated in response to input to Internet-based web pages. Examples include the Yellow Pages, http://uswestdex.com, the American Business Information database, http://infousa.com, and the American Society of Travel Agents, http://astanet.com.

In one new type of Internet-based marketplace, in which lists of merchants suitable for submission of requests for quotes by consumers is provided to the consumers based on consumer preferences and, where appropriate, on geographical proximity, a database of vendor information having certain specific features and characteristics is most desirable. The database of vendor information needs to be comprehensive and geographically broad-based with regard to vendors, so that any particular consumer has a high likelihood of being matched to several vendors that offer a particular good or service desired by the consumer and that are located

within a reasonable distance from the consumer's residence. The vendor database needs to contain the business name and zip code for each vendor, along with indications of the categories and subcategories of goods and services offered by the vendor. The vendor database must contain sufficient information to uniquely identify each vendor, such as the vendor's address, phone number, and email address. Finally, the vendor database needs to contain, for each vendor, a pointer, or method of contact, for submitting requests for quotes, such as email addresses, online form pointers or links, or fax numbers.

Unfortunately, currently-available databases do not contain all the different types of vendor information needed for the above-mentioned new type of Internet-based market. Many directories of businesses do not contain complete contact information and do not contain detailed information about the goods and services offered by vendors. Other databases that focus on particular types of businesses, such as the American Society of Travel Agents database mentioned above, may contain much of the detailed information required, but lack the depth of content needed by the Internet-based market. A more serious deficiency of many of the currently-available databases is that they require vendors to actively seek to be listed by the organization that compiles and maintains the database. However, in the Internet-based marketplace, it is more desirable to identify vendors from many types of available information, including vendor homepages and web-based references to vendors, so that the broadest possible compilation of vendors can be obtained.

The new, Internet-based market, described above, thus needs a vendor database compiled and organized to include sufficient information for the generation of lists of merchants suitable to particular consumers based on consumer preferences and geographical proximity. The vendor database needs also to be organized to allow for highly automated, continuous update that provides for extracting vendor from a variety of web-based information sources and merging the extracted information into the vendor database. Automated vendor information extraction and database update provide the only cost-effective means for obtaining the breadth of content required by the Internet-based market.

Summary of the Invention

The vendor database component of one embodiment of the present invention is implemented as a relational database. This vendor database component comprises a collection of relational tables that include tables containing detailed vendor information, tables presenting a many-to-many relationship between vendors and categories and subcategories of goods and services offered by the vendors, and a number of tables that facilitate extraction of new vendor information from web-based information sources and merging that extracted vendor information into the above-mentioned vendor information and category tables. The present invention relates both to a vendor database, as well as to a method and system for the highly automated construction, maintenance, and enhancement of the vendor database, including web-based data extraction tools, merging tools, and database construction and update processes.

Brief Description of the Drawings

Figure 1 shows a hypothetical homepage for a manufacturer's association.

Figure 2 represents the displayed list of vendors resulting from entry of the zip code "98104" into the text entry box of the homepage displayed in Figure 1.

Figure 3 depicts the link page referenced by the hyperlink in Figure 1.

Figure 4 represents the web page referenced by the first hyperlink in Figure 3.

Figure 5 represents the web page referenced by the second hyperlink in Figure 3.

Figure 6 represents the web page referenced by the third hyperlink in Figure 3.

Figure 7 is a flow control diagram for the continuous vendor database construction and update processes of the present invention.

Detailed Description of the Invention

The present invention is related to a new type of technology-enabled marketplace that is described in detail in U.S. Patent Application No. _____, entitled "Intelligent Multi-Media Market," assigned to the Imandi Corporation, and filed on January 15, 1999 that is hereby incorporated by reference in its entirety. The intelligent multi-media market ("IMMM") facilitates quote submissions from vendors to potential customers by providing to potential customers lists of vendors that offer a particular good or service desired by the potential customer, that meet certain requirements and preferences specified by the potential customer, and that, where appropriate, are located within a reasonable geographic distance from the customer's residence. An important component of the IMMM is a vendor database that is organized to contain sufficient information about vendors to enable the IMMM to construct lists of vendors appropriate for submission of a request for quotes from a potential customer seeking a particular good or service and that can be constantly updated through highly automated processes in order to contain very comprehensive, broad-based vendor information.

The present invention will be described in detail in three subsections that follow. In the first subsection, the present invention will be presented in overview via a series of illustrative examples. In the second subsection, a detailed schema of the vendor database is provided. In a third subsection, flow control diagrams and pseudocode descriptions of data extraction and data merging tools are provided.

Overview

In this subsection, the present invention is described in overview using simplified relational tables and a relatively simple data extraction example. The relational tables provided in the current subsection are simplified versions of like-named relational tables described below, in the following subsection, in which the entire database schema is presented. This subsection is intended to illustrate the overall database construction and data extraction processes as well as the nature of information stored in the various relational tables that together compose the vendor database and interrelationships between the data stored within the vendor database.

One organizing principle of the vendor database of the present invention is a comprehensive categorization and sub-categorization of different goods and services provided by vendors. Table 1, below, contains a brief textual description of some representative categories and subcategories of goods and services along with numerical identifiers that uniquely identify categories and subcategories, and also contains standard industrial classification ("SIC") numbers for those categories and subcategories classified by the SIC classification system. A list of SIC classifications is available at http://www.theodora.com/sic_b1.html. Table 1 is not a vendor database table, but is presented below to illustrate categories and subcategories, cross-indexed with SIC numbers, that will be used in sample data provided in the various relational tables that follow.

Table 1

Category	Category Number	Subcategory	Subcategory Number	SIC
auto	436			5511
auto	436	new Fordge	806	
auto	436	new Datsoya	124	
auto	436	used Fordge	807	
auto	436	used Datsoya	125	
brakes	1116			
brakes	1116	manufacture	307	3714
brakes	1116	repair	17	7534
brakes	1116	retail	377	3714
brakes	1116	wholesale	378	3714
copy	7132			
copy	7132	service	116	7334
copy	7132	machines	1844	5004
copy	7132	machine repair	1900	
books	26			
books	26	retail general	88	5942
books	26	retail technical	104	5942
books	26	repair	119	
books	26	wholesale/ distributor	188	5192
confections	8			
confections	8	chocolate covered ants	13	
confections	8	chocolate covered insects	11	
confections	8	chocolate bars	44	5411
confections	8	hard candy	49	5441
confections	8	chocolate covered crustaceans	7	
confections	8	chocolate covered echinoderms	9	

Referring to Table 1, there is a category "auto," having category number "436," that is cross-indexed by the SIC number "5511." Within the category "auto" are subcategories related to the retail sale of new and used Fordge and Datsoya automobiles, each having a unique subcategory number. The vendor database category and subcategory classification system may be far more detailed and complex than the SIC classification system, and may be, in part, generated automatically during the data extraction and database update phases of the present invention.

Table 2, below, is a representation of the relational table "Company." This table contains basic information about each identified vendor.

Table 2

Company (simplified)

Company Id	Name	Street Address	City	State	Zip	Email	Phone
36	Big Bad Auto	911 James	Corvalis	OR	97012	bad@bad.com	503 777 1111
94	Jerry's Datsoya	1183 Wood	Smallville	CA	94082	jerrys@p12.com	209 866 7322
113	Elvis Brake	876 Main	Centertown	KA	51032	elvis@kabrake.com	785 934 8106
119	Bob's Copy	Suite 11B 841 First St	Seattle	WA	98104	bobs@copy.com	206 341 1716
136	Eric's Fordge	1101 Greenlake Way	Seattle	WA	98151	eric@tap.com	206 771 1300
147	Anita's Books	710 Baker	Wicksville	NY	20122	anita@wibook.com	315 812 7689
217	Beth's Books	333 Cherry	Zaksville	TN	31261	beth@zak.com	615 548 1542
222	Downtown Ants	600 Pine	Seattle	WA	98104	dta@a.com	206 340 9578

As discussed below in the following subsection, the table "Company" in the preferred embodiment contains many more columns than the simplified version of the table "Company," shown above. However, for the present example, the simplified tables, such as the table "Company," shown above, provide a clear illustration of the nature of data and the interrelationships between the data, stored within the vendor database

relational tables. Note also that the above-referenced U.S. Patent Application No. _____ presents a different, alternate embodiment of the vendor database. The present embodiment is especially well suited for construction and continuous update by automated methods, to be described below.

Each row in the relational table "Company" includes the following fields, corresponding to columns of the relational table "Company," shown above as Table 2: (1) "CompanyID," a unique numerical identifier for each vendor; (2) "Name," the name of the vendor; (3) "StreetAddress," the location of the vendor, specified in the simplified table "Company" as a single postal address; (4) "City," the city in which the vendor is located; (5) "State," the 2-character abbreviation for the state in which the vendor is located; (6) "Zip," the zip code corresponding to the vendor's location specified in the previous two fields; (7) "Email," the Internet email address at which the vendor can be contacted; and (8) "Phone," a phone number for the vendor. The table "Company," shown above, contains eight example rows specifying a number of different vendors. As specified in the following subsection, the table "Company" may contain a number of different addresses and phone numbers for a particular vendor as well as additional information. When employed in a functioning IMMM, the table "Company" generally contains many tens or hundreds of thousands of rows.

The relational table "CategoryMap," shown below, represents a many-to-many relationship between vendors and category and subcategory business classifications. Example category and subcategory business classifications are shown above, in Table 1.

Table3

CategoryMap

CompanyId	CategoryId	SubcategoryId
36	436	806
36	436	807
94	136	124
94	136	125
113	1116	17
119	7132	116
136	436	806
136	436	807
147	26	58
147	26	104
217	26	88
222	8	13

Each row, or entry, in the relational table "Category Map" contains the following field corresponding to columns of the table: (1) "CompanyId," the unique numerical identifier for a vendor; (2) "CategoryId," a numerical identifier of a category of business; and (3) "SubcategoryId," a numerical identifier of a business sub-category. For example, the sample data in the first row of the above-shown relational table "CategoryMap" indicates that the vendor identified by CompanyId "36," the vendor "Big Bad Auto" described by the first entry in Table 2, sells new Fordge automobiles, where the CategoryId "436" corresponds to automobiles and the SubcategoryId "806" corresponds to new Fordge automobiles, as seen above in Table 1.

The relational table "CompanyMerge," shown below in two parts, contains information concerning the data source and a time stamp for each field of each row in the relational table "Company," shown above in Table 2.

Table 4

CompanyMerge (simplified) Columns 1-8

Company Id	Name Source	Name Date	StreetAdd Source	StreetAdd Date	City Source	CityDate	State Source
36	13	12/13/98 1:12 PM	13	12/13/98 1:12 PM	13	12/13/98 1:12 PM	13
94	13	12/13/98 1:50 PM	13	12/13/98 1:50 PM	13	12/13/98 1:50 PM	13
113	67	1/4/99 12:31 PM	67	1/4/99 12:31 PM	67	1/4/99 12:31 PM	67
119	85	1/25/99 8:05 AM	85	1/25/99 8:05 AM	85	1/25/99 8:05 AM	85
136	13	12/13/98 1:48 PM	13	12/13/98 1:48 PM	13	12/13/98 1:48 PM	13
147	21	1/13/99 8:25 AM	21	1/13/99 8:25 AM	21	1/13/99 8:25 AM	21
217	21	1/13/99 8:50 AM	21	1/13/99 8:50 AM	21	1/13/99 8:50 AM	21
222	17	1/10/99 11:50 AM	17	1/10/99 11:50 AM	17	1/10/99 11:50 AM	17

Table 5

CompanyMerge (simplified) Columns 9-15

State Date	Zip Source	Zip Date	Email Source	Email Date	Phone Source	Phone Date
12/13/98 1:12 PM	13	12/13/98 1:12 PM	13	12/13/98 1:12 PM	13	12/13/98 1:12 PM
12/13/98 1:50 PM	13	12/13/98 1:50 PM	13	12/13/98 1:50 PM	13	12/13/98 1:50 PM
1/4/99 12:31 PM	67	1/4/99 12:31 PM	67	1/4/99 12:31 PM	67	1/4/99 12:31 PM
1/25/99 8:05 AM	85	1/25/99 8:05 AM	85	1/25/99 8:05 AM	85	1/25/99 8:05 AM
12/13/98 1:48 PM	13	12/13/98 1:48 PM	13	12/13/98 1:48 PM	13	12/13/98 1:48 PM
1/13/99 8:25 AM	21	1/13/99 8:25 AM	21	1/13/99 8:25 AM	21	1/13/99 8:25 AM
1/13/99 8:50 AM	21	1/13/99 8:50 AM	21	1/13/99 8:50 AM	21	1/13/99 8:50 AM
1/10/99 11:50 AM	17	1/10/99 11:50 AM	17	1/10/99 11:50 AM	17	1/10/99 11:50 AM

The first 8 columns of the relational table "CompanyMerge" are shown above, in Table 4, and the latter 7 columns of the relational table "CompanyMerge" are shown above in Table 5. The table is split into two parts for the sake of clarity of display.

Each row, or entry, in the relational table "CompanyMerge" contains the following fields, corresponding to columns of the table: (1) "CompanyId," the unique numerical identification of a vendor; (2) "NameSource," a numeric identification of the data source from which the name of the vendor is extracted by the automated data extraction process, to be discussed below; (3) "NameDate," a date and time stamp indicating the time at which the name of the vendor was extracted from the data source, identified in the previous field, by the automated data extraction process; and (4) similar source and date fields for each of the columns of the relational table "Company" that follow the column "Name," with the data source and time of data extraction indicated in the source and date fields of the relational table "CompanyMerge" for each field of the relational table "Company." The relational table "CompanyMerge" is used during the merging process for merging newly found information about vendors into the relational table "Company." By comparing the data source and time of extraction of new information to the data and time of extraction of information already entered into the relational table "Company," the merging process, to be discussed below, can ensure that only information more current than, or more reliable than, information already resident in the relational table "Company" will be merged into the relational table "Company" during the merging process.

The relational table "MergeCategory," shown below, contains data source and data extraction time information about the entries in the relational table "CategoryMap" in the same way that the relational table "CompanyMerge" contains data source and extraction time information about the data stored in the relational table "Company," as described in the preceding paragraph.

Table 6

MergeCategory (simplified)

Company Id	Category Id	Subcategory Id	Source	Date
36	436	806	13	12/13/98 1:12 PM
36	436	807	13	12/13/98 1:12 PM
94	136	124	13	12/13/98 1:50 PM
94	136	125	13	12/13/98 1:50 PM
113	1116	17	67	1/4/99 12:31 PM
119	7132	116	85	1/12/99 8:25 AM
136	436	806	13	12/13/98 1:48 PM
136	436	807	13	12/13/98 1:48 PM
147	26	58	21	1/13/99 8:25 AM
147	26	104	21	1/13/99 8:25 AM
217	26	88	21	1/13/99 8:50 AM
222	8	13	17	1/10/99 11:50 AM

Each row or record, of the relational table "MergeCategory" contains the following fields: (1) "CompanyId," the unique numerical identification of a vendor; (2) "CategoryId," a numerical identification of a business category; (3) "SubcategoryId," a numerical identification of a business subcategory; (4) "Source," a numerical identifier of the data source that led to the CompanyId/CategoryId/SubcategoryId entry indicated by the contents of the preceding fields; and (5) "Date," the date and time that the data was extracted form the data source identified in the preceding field. Thus, the relational table "MergeCategory" contains data source and extraction time information for each entry in the relational table "CategoryMap." As with the information in the relational table "CompanyMerge," described above, this information is used during the data merging process, to be described below.

Three indexes contained in the relational tables, "NameIndex," "AddressIndex," and "PhoneIndex" follow.

Table 7

AddressIndex

CompanyId	AddressToken	State	Ordinal
36	911	OR	1
36	james	OR	2
94	1183	CA	1
94	wood	CA	2
113	876	KA	1
113	main	KA	2
119	suite_116	WA	1
119	841	WA	2
119	1st	WA	3
136	1101	WA	1
136	greenlake	WA	2
147	110	NY	1
147	baker	NY	2
217	333	TN	1
217	cherry	TN	2
222	600	WA	1
222	pine	WA	2

Table 8

NameIndex

CompanyId	NameToken	State	Ordinal
36	big	OR	1
36	bad	OR	2
36	auto	OR	3
94	jerrys	CA	1
94	datsoya	CA	2
113	elvis	KA	1
113	brake	KA	2
119	bobs	WA	1
119	copy	WA	2
136	erics	WA	1
136	fordge	WA	2
147	anitas	NY	1
147	books	NY	2
217	beths	TN	1
217	books	TN	2
222	downtown	WA	1
222	ants	WA	2

Table 9

PhoneIndex

CompanyId	Phone	State
36	7771111	OR
94	8667322	CA
113	9348106	KA
119	3411716	WA
136	7711380	WA
197	8127689	NT
217	5481542	TN
222	3409578	WA

These relational tables are used to store tokenized portions of certain of the data fields of the relational Table "Company" to serve as lookup indexes for matching newly

extracted information to existing vendor information in the relational table "Company." Each entry, or row, in the relational table "NameIndex," shown above, contains the following fields, corresponding to columns of the table: (1) "CompanyId," the unique numerical identification of a vendor; (2) "NameToken," a tokenized portion of the contents of the field "Name" in the relation table "Company" for the row in the relational table "Company" having the same CompanyId value as the value stored in the preceding field; (3) "State," the state in which the vendor is located, used for partitioning the index in order to speed lookup; and (4) "Ordinal," the position of the tokenized portion of the vendor name contained in the field "NameToken" within the text representation of the vendor name in the field "Name" of the relational table "Company". For example, referring to Table 2, the name of the vendor having CompanyId "36" is "Big Bad Auto." Tokenization of this name results in the following three tokens: "big," "bad," and "auto." These three tokens are found in the first three rows of the relational table "NameIndex" with ordinal values "1," "2," and "3," respectively The relational table "AddressIndex" is analogous to the relational table "NameIndex," except that tokenized portions of the vendors' addresses, tokenized from the contents of the field "StreetAddress" in the relational table "Company," are stored in the field "AddressToken" of the relational table "AddressIndex." As with the relational table "NameIndex," the relational table "AddressIndex" is used to match newly extracted information from various web sites to existing vendor entries in the relational table "Company." Finally, the relational table "PhoneIndex" contains tokenized versions of the vendor phone numbers and serves as an additional index in analogous fashion to relational tables "NameIndex" and relational table "AddressIndex."

The above-described relational tables shown in Tables 2-9 together comprise the core of the vendor database. These tables contain information about vendors, including a mapping between vendors and business categories and subcategories, information concerning the source and extraction time for each portion of each vendor record and indexing information to allow newly extracted information to be correlated with the information contained in Tables 2 and 3. These tables, and the sample data contained in them, will now be used in a detailed example to illustrate the processes of vendor database construction and update.

Information about new vendors note currently listed in the relational table "Company," and more current information about vendors already represented by entries in the relational table "Company," can be obtained from a variety of Internet-based information sources. The automated data extraction tools and methods of the present invention are focused on extracting information from web pages, although the methodology can be applied to many different types of information sources with slight modifications. Figure 1 shows a hypothetical homepage for a manufacturers' association. On the homepage 102 of Figure 1, a reader can type the reader's zip code into a text entry box 104 and then press the return key of the personal computer ("PC") on which the reader is viewing the homepage in order to see a display of all retail vendors of the manufacturers' product located within a reasonable distance of the reader.

Figure 2 represents the displayed list of vendors resulting from entry of the zip code "98104" into the text entry box of the homepage displayed in Figure 1. Two vendors are listed in Figure 2, each vendor represented by a row 202 and 204 within a four-column table 206. The homepage displayed in Figure 1 provides a means for obtaining listings of vendor information that can be readily extracted and transformed into entries in the relational table "Company." Thus, the homepage depicted in Figure 1 serves as an information portal into a specific vendor database to which access is offered via the Internet. Such portals to existing databases are commonly identified by IMMM technicians and, if the database is determined to be sufficiently large to warrant preparation of a data extraction tool, a data extraction tool is prepared by one or more IMMM technicians in order to automatically extract the contents of the database and prepare IMMM vendor database records corresponding to the extracted information. In the case of the present example, the data extraction tool may be written as a small Java program running within the context of a web browser.

The data extraction tool navigates to the homepages shown in Figure 1 and extracts and processes as much vendor information as possible using the search mechanism supported by the web page. In this example, the data extraction tool might either generate all possible zip codes, or select zip codes from a previously-prepared list of zip codes and, one-by-one, enter the generated or selected zip codes into the text entry box 104 and process resulting lists of vendors, such as the list displayed in Figure 2.

Processing the lists of vendors involves parsing the contents of the HTML files that represent each displayed vendor list. Because the vendor information is presented in a regular format, the data extraction tool can parse out the zip code, city, and state for the merchants displayed on a single page from the header information displayed on that page 208 and then parse out additional vendor information from the fields within each successive row of the four-column table 206. Thus, the data extraction tool can easily extract information about all vendors listed in the database from which the vendor listings, such as the vendor listing of Figure 2, are programmatically generatedby the manufacturers' web site.

The data extraction tool places newly extracted vendor information into a separate relational database table, table "InterDB1." This information is placed into a separate table because the newly extracted vendor information must be compared to the contents of relational table "Company" before the newly extracted information can be entered into relational table "Company," by a merging tool. First, the merging tool determines whether the vendor represented by the newly extracted vendor information is already described by an entry in the IMMM vendor database. If not, and if the newly extracted data information is relatively complete, a new IMMM vendor database record can be constructed within the relational table "Company" to represent the newly identified vendor, and information copied into the newly constructed vendor database record from relational table "InterDV1." However, if the newly extracted information relates to a vendor that is already present in the IMMM vendor database, then the newly extracted information may be used by the merging tool to update an existing entry in the IMMM vendor database provided that the newly extracted information is either more current, or has been extracted from a data source that is deemed to be more reliable or of higher quality by the merging tool. In the case of incomplete or ambiguous data, the data may need to be verified and analyzed by a human technician, who then decides whether the data is sufficiently complete to justify creation of a new IMMM vendor database record, whether the data relates to a vendor already described in the IMMM database, or whether the data should simply be discarded. Thus, to facilitate the above-mentioned merging process, newly extracted vendor information is placed into the separate relational table "InterDB1," shown below, by data extraction processes.

Table 10

InterDBI (simplified)

Category Id	Sub category Id	Name	Street Address	City	State	Zip	Email	Phone
8	13	Downtown Ants	520 Pike Street	Seattle	WA	98104	dta@a.com	206 340 9578
8	13	Corner Ants	701 5th Avenue	Seattle	WA	98104	cant@s.com	206 622 4901
8	13	Chocolate Covered Ants Trade Association	701 5th Avenue	Seattle	WA	98104	chocolate_ants@cants.com	800 825 4626
8	13	Brenda's Chocolate Covered Managerie	3702 State Street	Madison	WI	53708	brendas@madtown.com	608 624 1495
8	11	Brenda's Chocolate Covered Managerie	3702 State Street	Madison	WI	53708	brendas@madtown.com	608 624 1495
8	7	Brenda's Chocolate Covered Managerie	3702 State Street	Madison	WI	53708	brendas@madtown.com	608 624 1495
8	9	Brenda's Chocolate Covered Managerie	3702 State Street	Madison	WI	53708	brendas@madtown.com	608 624 1495

The relational table "InterDB1" has the same columns as the relational table "Company," although any particular entry in the relational table "InterDB1" may not have values for the columns, depending on the completeness of the newly extracted data. For the current example, information extracted and parsed from the vendor list shown in Figure 2 has been entered into the first two rows of the relational table "InterDB1."

Referring back to Figure 1, there is a hyperlink 106 entitled "Links" at the bottom of the manufacturers' association homepage. When the reader positions a cursor over this hyperlink and depresses a mouse button, the reader's browser displays a linkpage with hyperlinks to related web pages. Figure 3 depicts the link page corresponding to the hyperlink in Figure 1. Linkpages may contain tens or hundreds of links to a variey of different types of pages. In the current example, the linkpage contains three links 302, 304, and 306. When the reader positions a cursor over a hyperlink shown in Figure 3, the reader's browser then displays a new web page referenced by the hyperlink. Figure 4 represents the web page referenced by the first hyperlink in Figure 3. Figure 5 represents the web page referenced by the second hyperlink in Figure 3. Figure 6 represents the web page referenced by the third hyperlink in Figure 3.

In general, chains of hyperlinks represent a mathematical graph superimposed over a collection of web pages. Any two nodes, or web pages, within the

graph may be joined by one or more edges, or hyperlinks, and any node, particularly link pages, such as the linkpage shown in Figure 3, may contain numerous hyperlinks directed to other web pages. There are generally many different possible paths by which the graph can be navigated to touch each node, and arbitrarily selected paths may contain redundancies, cycles, and endless loops. Although a human technician may be capable of slowly traversing the nodes within a hyperlink graph, large-scale data extraction from hyperlink graphs is best accomplished by automated means. For IMMM vendor database construction and update, web crawler tools are used to extract information from hyperlink graphs. The web crawler tools start at a given web page and then follow links and parse information displayed on linked web pages in order to identify and extract new vendor information.

Continuing with the current example, once a data extraction tool has extracted the merchant database information by supplying zip codes to homepages, displayed in Figure 1, a web crawler data extraction tool can then be directed to start extracting information from the graph of web pages referenced by the hyperlink 106 on the homepage. The web crawler data extraction tool first navigates to the linkpage displayed in Figure 3, and from that page to each of the web pages displayed in Figures 4-6. When the web crawler tool arrives at the web pages shown in Figure 4, the web crawler tool attempts to identify vendor information displayed on the page. The web crawler tool looks for character strings representative of company names, addresses, phone numbers, email addresses, zip codes, and other information contained in the fields of relational tables "Company" and "InterDB1." In the case of Figure 4, the web crawler data extraction tool determines that no vendor information is displayed on the page. Although some of the words might be construed by the web crawler data extraction tool to represent the name of a vendor, there is not sufficient additional information, such as addresses or phone numbers, within close proximity to any tentative vendor name in order to justify an attempt to extract vendor information from the page.

The web page displayed in Figure 5 represents information of intermediate interest to the web crawler data extraction tool. The web crawler data extraction tool may identify the first line in Figure 5 502 as the name of a vendor and may identify the email address 504 and phone number 506 as the email and phone

number for the vendor. Although the vendor information is incomplete, the web crawler data extraction tool, in this case, may determine that there is sufficient information to enter the extractable data into the relational table "InterDB1" as an incomplete vendor record, shown above in row 3 of Table 10. Finally, when the web crawler data extraction tool arrives at the web pages shown in Figure 6, the web crawler data extraction tool finds sufficient information on the page to construct the tentative vendor database records contained in rows 4-7 of Table 10, above. Note that the web crawler data extraction tool not only identifies vendor information, but also potential business categories and subcategories, thus constructing, in this case, four records corresponding to four different identified business category/subcategory pairs having CategoryId "8" and SubcategoryId's "13," "11," "7," and "9." As can be seen in Table 1, above, these category/subcategory pairs correspond to the following business categories: chocolate covered ants, chocolate covered insects, chocolate covered crustaceans, and chocolate covered echinoderms.

The data extraction tools and web crawler data extraction tools can be run periodically or continuously to build up multiple tables of potential vendor information, such as relational table "InterDB1" shown above as Table 10. These intermediate tables of potential vendor records are then processed by a merging tool. A merging tool analyzes each record, or row, within the intermediate tables in order to either create, based on that row, a new IMMM vendor database record, update an existing IMMM vendor database record, or flag the data contained in the intermediate table row for analysis by a human technician. The merging process can be described with reference to the first record of the relational table "InterDB1," above. First, the merging tool tokenizes the contents of the following columns: "Name," "StreetAddress," and "Phone." As discussed above, these three different pieces of information may be tokenized in different ways. In the case of the contents of the field "Name," tokenization produces the tokens "downtown" and "ants." Tokenization of the contents of the field "StreetAddress" produces the single token "520 pike." Tokenization of the phone number produces the single token "3409578." The merging tool then uses these prepared tokens to find matching entries in the relational tables "NameIndex," "AddressIndex," and "PhoneIndex," displayed above in Tables 7-9. Each

identified match is scored and entered into the relational table "tempdb.dbo.TokenMatch," shown below:

Table 11

tempdb.dbo.TokenMatch

CompanyId	Score
222	20
222	30
222	20

Each row in relational table "tempdb.dbo.TokenMatch" contains the following two fields: (1) "CompanyId," the unique numerical identifier that specifies a vendor company with a matching token entry in Tables 7, 8, or 10; and (2) "Score," an empirical rating of the quality of the match represented by the entry. The three entries in Table 11 correspond to matches of the tokens "downtown" and "ants" with the last two rows of the relational table "NameIndex" and the token "3409578" with the last row of relational table "PhoneIndex." In the present example, all three matches relate to the vendor identified by CompanyId "222." However, in a live, functioning IMMM vendor database, hundreds of matches related to many different companies may be detected and entered into the relational table "tempdb.dbo.TokenMatch.". The merging tool next computes a cumulative total score for each CompanyId having entried in the relational table "tempdb.dbo.TokenMatch" and enters that total score, along with the corresponding CompanyId, into the relational table "tempdb.dbo.TokenSummary," shown below in Table 12:

Table 12

tempdb.dbo.TokenSummary

CompanyId	TotalScore
222	70

Again, in general, relational table "tempdb.dbo.TokenSummary" may contain tens or hundreds of different entries following the analysis of a single record from the relational table "InterDb1" by the merging tool. The merging tool selects the CompanyId or

CompanyIds from the relational table "tempdb.dbo.TokenSummary" associated with the highest total score. In the current example, there is only one entry containing the total score "70" associated with the CompanyId "222." Thus, in the present example, the first record of the relational table "InterDB1" is identified by the merging tool as corresponding to the existing vendor record in the relational table "Company" identified by CompanyId "222."

At this point, the merging tool then does a field-by-field comparison of the information in the selected row of the relational table "InterDB1" with the row in the relational table "Company" identified, via token matching, as describing the vendor represented by the selected row of the relational table "InterDB1." This field-by-field comparison is facilitated by the entry in the relational table "CompanyMerge" having the same value for the field "CompanyId" in the relational table "CompanyMerge" as the value stored in the field "CompanyId" of the identified row of the relational table "Company." Thus, the field-by-field comparison conducted by the merging tool, in the present example, concerns the first row of relational table "InterDB1," the last row in the relational table "Company," and the last row in the relational table "CompanyMerge." First, the merging tool compares the fields "Name" in the relational tables "InterDB1" and "Company." In both tables, the value contained in the field "Name" is identical. Thus, no update is needed. A field-by-field comparison, proceeding through each of the remaining columns in the relational table "InterDB1," reveals only a single difference between the contents of the corresponding fields of the relational tables "Company" and "InterDB1": the value for the field "StreetAddress" in the relational table "Company" is "600 Pine" while the value for the field "StreetAddress" in the relational table "InterDB1" is "520 Pike Street." Thus, the merging tool determines that the StreetAddress for the vendor "Downtown Ants" may have changed. To determine whether or not to update the field "StreetAddress" in relational table "Company," the merging tool consults the relational table "CompanyMerge" to determine the data source and extraction date for the information contained in the field "StreetAddress" in relational table "Company." Then, the merging tool compares the data source to the data source from which the data in the relational table "InterDB1" was extracted, in the case that the data represents data extracted from a single source, or, alternatively, uses the value from a data source

column (not shown) in the relational table "InterDB1" when more than one data source is used to construct relational table "InterDB1," and compares the time stamp for the information contained in the field "StreetAddress" of relational table "Company" to the current date and time. If the data source from which the entry in the relational table "InterDB1" is deemed by the merging tool to be as reliable as, or more reliable, than the data source from which the original street address was obtained by the relational table "Company," or if the merging tool determines that the information stored in the field "StreetAddress" of relational table "Company is less timely than the information stored in the field "StreetAddress" of the relational table "InterDB1," then the merging tool will copy the value for the street address stored in relational table "InterDB1" into the field "StreetAddress" of the relational table "Company." Thus, by continuously extracting data from web pages and processing the extracted data with the merging tool, the IMMM vendor database is constantly updated in order to contain the most reliable and most timely information.

When the merging tool analyzes the second record in the relational table "InterDB1," no matches of tokens are found in Tables 7-9. The merging tool therefore creates a new row, or record, in the relational table "Company" and copies values from the field in the second row of relational table "InterDB1" into the newly created record. Continuous data extraction and processing of extracted data by the merging tool ensures an increasingly comprehensive and broad-based compilation of vendor information in the IMMM vendor database.

When the merging tool processes the third row of the relational table "InterDB1," the merging tool may determine that insufficient information is present within the record, or, in other words, too many fields within the record have NULL values. Instead of attempting to match the record to existing vendor records, or, upon attempting to match the incomplete record with existing vendor records and finding no total score within the relational table "tempdb.dbo.TokenSummary" higher than some threshold value, the merging tool determines that the record is ambiguous and places the ambiguous record into the relational table "CompanyAmbiguous," shown below:

Table 13

CompanyAmbiguous (simplified) Columns 1-8

Id	Name	Street Address	City	State	Zip	Email	Phone
97	Chocolate Covered Ants Trade Association					chocolate_ants @ccants.com	800 825 4626

Table 14

CompanyAmbiguous (simplified) Columns 9-17

CategoryId	SubCatId1	CategoryId2	SubCatId2	CategoryId3	SubCatId3
8	13				

The relational table "CompanyAmbiguous" contains columns equivalent to the columns of the relational tables "Company" and "CategoryMap." The relational table "CompanyAmbiguous" stores newly extracted but ambiguous information that must be subsequently presented to an IMMM human technician for evaluation. Again, as discussed above, the extracted information may end up in the relational table "CompanyAmbiguous" if the extracted information is fragmentary and incomplete, but indicative of vendor information, or if the highest total score in the relational table "tempdb.dbo.TokenSummary" resulting from the matching process is greater than some lower threshold value, indicating a potential match to one or more existing vendor records, but lower than a higher threshold value that represents the threshold for a definite match to one or more existing IMMM vendor database records. A web browser-based verification tool can be employed to graphically display each record from the relational table "CompanyAmbiguous," along with potentially matching records from the relational table "Company," to allow an IMMM human technician to quickly and visually compare the newly extracted, but ambiguous information, with existing IMMM vendor database records and to decide whether to enter the newly extracted information into the vendor database as a new record or an update to an existing record.

Note that the merging tool, in similar fashion, merges newly detected business category information for previously identified and newly identified vendors

into the relational table "CategoryMap" using information stored in the relational table "MergeCategory" during the merging process. Thus, for example, the final four records in the relational table "InterDB1" will result in a new record for the vendor "Brenda's Chocolate Covered Menagerie" in the relational table "Company" as well as four new entries in the relational table "CategoryMap" that indicate a relationship between the vendor "Brenda's Chocolate Covered Menagerie" and the business category "Confections" and business subcategegories "Chocolate Covered Ants," "Chocolate Covered Insects," "Chocolate Covered Crustaceans," and "Chocolate Covered Echinoderms." Updated versions of the relational tables "Company" and "CategoryMap," shown below, represent the new contents of these tables following data extraction and merging based on the above examples.

Table 15

Company (simplified)

Company Id	Name	Street Address	City	State	Zip	Email	Phone
36	Big Bad Auto	911 James	Corvalis	OR	97012	bad@bad.com	503 777 1111
94	Jerry's Datsoya	1183 Wood	Smallville	CA	94082	jerrys@p12.com	209 866 7322
113	Elvis Brake	876 Main	Centertown	KA	51032	elvis@kabrake.com	785 934 8106
119	Bob's Copy	Suite 11B 841 First St	Seattle	WA	98104	bobs@copy.com	206 341 1716
136	Eric's Fordge	1101 Greenlake Way	Seattle	WA	98151	eric@tap.com	206 771 1300
147	Anita's Books	710 Baker	Wicksville	NY	20122	anita@wibook.com	315 812 7689
217	Beth's Books	333 Cherry	Zaksville	TN	31261	beth@zak.com	615 548 1542
222	Downtown Ants	520 Pike Street	Seattle	WA	98104	dta@a.com	206 340 9578
223	Corner Ants	701 5th Avenue	Seattle	WA	98104	cant@s.com	206 622 4901
224	Brenda's Chocolate Covered Manageric	3702 State Street	Madison	WA	53708	brendas@madtown.com	603 624 1495

Table 16

CategoryMap

CompanyId	CategoryId	SubcategoryId
36	436	806
36	436	807
94	436	124
94	436	125
113	1116	17
119	7132	116
136	436	806
136	436	807
147	26	58
147	26	104
217	26	88
222	8	13
223	8	13
224	8	13
224	8	11
224	8	9
224	8	7

It should be noted that the indexing tables, Tables 7-9, are also automatically updated by the merging tool, as are the relational tables "CompanyMerge" and "MergeCategory." Thus, for example, when the address of vendor "Downtown Ants" is updated in the above example, the data source and time stamp for the field "StreetAddress" are updated in the relational table "CompanyMerge" for the record identified by the CompanyId "222." Similarly, the existing records in the relational table "AddressIndex" identified by the CompanyId "222" are removed from the table, and new records having AddressToken values of "520" and "pike" and having CompanyId values of "222" are added to the table.

As noted above, the tables described in the following subsection contain additional columns representing additional types of vendor information, and the database schema includes additional tables. Examples presented in this subsection illustrate the overall process of vendor data extraction and merging and illustrates the core data, core tables, and important interrelationships between the different types of data stored in the core tables of the IMMM vendor database. As noted above, the

IMMM vendor database described in the current application is designed specifically for automated data extraction and merging tools that run on a relatively continuous basis in order to create, maintain, and update the IMMM vendor database. This embodiment differs from the embodiment described in U.S. Patent Application No._____, referenced above. The database described in that application is tailored for ease of implementation and with special regard to form generation. Other, alternative embodiments of the IMMM vendor database may be developed to enhance efficiency related to other particularized aspects of the IMMM. A particular instantiation of the IMMM may employ any one of these embodiments, or a combination of these embodiments either in a single database, or in multiple databases.

Database Schema

In the present subsection, the database schema for the IMMM vendor database of the present invention is provided in a series of SQL-like "CREATE" statements. Simplified versions of many of these tables are illustrated, with sample data, in the previous subsection. Therefore, the descriptions presented in the current subsection will focus on those features and aspects of the database schema not illustrated and described in the previous subsection.

The three basic vendor information containing tables of the IMMM vendor database are the relational tables "Company," "CompanyExtra," and "CategoryMap." The SQL-like "CREATE TABLE" statements that instantiate these tables are provided below, along with an SQL-like "CREATE INDEX" statement that creates a clustered index on fields "CategoryId," "SubcategoryId," and "CompanyId" of the relational table "CategoryMap."

```
CREATE TABLE Company
(
    CompanyId       binary(8)       NOT NULL  PRIMARY KEY NONCLUSTERED,
    Name            varchar(255)    NOT NULL,
    StreetAddress   varchar(255)    NOT NULL,
    MailingAddress  varchar(255)    NULL,
    City            varchar(255)    NOT NULL,
    State           varchar(255)    NOT NULL,
    Zip             varchar(255)    NOT NULL,
    Email           varchar(255)    NULL,
    Web site        varchar(255)    NULL,
    Phone1          varchar(255)    NOT NULL,
    Phone2          varchar(255)    NULL,
    TollFree1       varchar(255)    NULL,
    TollFree2       varchar(255)    NULL,
    Fax             binary(8)       NOT NULL,
    ContactGender   tinyint         NULL,
    ContactFirst    varchar(255)    NULL,
    ContactMiddle   varchar(255)    NULL,
    ContactLast     varchar(255)    NULL,
    ContactPhone    binary(8)       NOT NULL,
    Active          bit             NOT NULL,
    QuoteWeight     smallint        NULL,
    ServerId        smallint        NOT NULL
)

CREATE TABLE CompanyExtra
(
    CompanyId       binary(8)       NOT NULL  PRIMARY KEY NONCLUSTERED,
    Longitude       float           NULL,
    Latitude        float           NULL,
    BillingData     varchar(255)    NULL
)

CREATE TABLE CategoryMap
(
    CompanyId       binary(8)   NOT NULL,
    CategoryId      int         NOT NULL,
    SubcategoryId   int         NULL,
    Active          bit         NOT NULL,
    QuoteWeight     smallint    NULL
)

CREATE UNIQUE CLUSTERED INDEX Category ON dbo.CategoryMap
        (CategoryId, SubcategoryId, CompanyId)
```

A simplified version of the relational table "Company" is illustrated and described in the previous subsection. The preferred embodiment of the relational table "Company," described by the SQL-like statement above, includes additional fields. For example, there is a "MailingAddress" field in addition to the "StreetAddress" field for storing the vendor's mailing address. The preferred embodiment provides both the field "Email" and the field "Web site" to store the email Internet address for sending email messages to the vendor and to store the Internet address of the vendor's home web page. The preferred embodiment has two fields for phone numbers, "Phone1," and "Phone2," and two fields for toll-free phone numbers, "TollFree1," and "TollFree2." In addition, the preferred embodiment has a field for a fax number, "Fax," as well as the following

fields that describe a human contact for the vendor: "ContactGender," "ContactFirst," "ContactMiddle," "ContactLast," and "ContactPhone." These fields describe the sex, first, middle, and last names, and the phone number of the human contact. The field "Active" in the preferred embodiment indicates whether or not the vendor described by an entry is actively receiving requests for quotes. The field "QuoteWeight" contains a numerical indication of the desirability of submitting requests for quotes to the vendor represented by the entry in the relational table "Company." The field "ServerId" is used for partitioning the IMMM vendor database across multiple server computers.

The relational table "CompanyExtra" contains additional information about a vendor identified by the contents of the field "CompanyId." The additional information includes the longitude and latitude of the vendor's location stored in the fields "Longitude," and "Latitude," as well as a textual description of billing information for the vendor stored in the field "BillingData."

The relational table "CategoryMap" is illustrated and described in the previous subsection, and the fields "Active" and "QuoteWeight," not described and illustrated in the simplified version of the relational table "CategoryMap" in the previous subsection, are analogous to the identically named fields in the relational table "Company," described above. The above three tables thus contain the essential vendor information that is used by the IMMM to produce lists of vendors that offer a particular good or service to IMMM clients seeking to receive quotes on that particular good or service.

The SQL-like "CREATE TABLE" statement for the relational table "CompanyAmbiguous," illustrated and described in the previous subsection, is provided below:

```
CREATE TABLE CompanyAmbiguous
(
    Id              int             IDENTITY(1, 1) PRIMARY KEY NONCLUSTERED,
    Name            varchar(255)    NOT NULL,
    StreetAddress   varchar(255)    NOT NULL,
    MailingAddress  varchar(255)    NULL,
    City            varchar(255)    NOT NULL,
    State           varchar(255)    NOT NULL,
    Zip             varchar(255)    NOT NULL,
    Email           varchar(255)    NULL,
    Web site        varchar(255)    NULL,
    Phone1          varchar(255)    NOT NULL,
    Phone2          varchar(255)    NULL,
    TollFree1       varchar(255)    NULL,
    TollFree2       varchar(255)    NULL,
    Fax             binary(8)       NOT NULL,
    ContactGender   tinyint         NULL,
    ContactFirst    varchar(255)    NULL,
    ContactMiddle   varchar(255)    NULL,
    ContactLast     varchar(255)    NULL,
    ContactPhone    binary(8)       NOT NULL,
    CategoryId1     int             NULL,
    SubCatId1       int             NULL,
    SIC1            int             NULL,
    CategoryId2     int             NULL,
    SubCatId2       int             NULL,
    SIC2            int             NULL,
    CategoryId3     int             NULL,
    SIC3            int             NULL,
    CategoryId4     int             NULL,
    SubCatId4       int             NULL,
    SIC4            int             NULL,
    CategoryId5     int             NULL,
    SubCatId5       int             NULL,
    SIC5            int             NULL,
    CategoryId6     int             NULL,
    SubCatId6       int             NULL,
    SIC6            int             NULL,
    CategoryId7     int             NULL,
    SubCatId7       int             NULL,
    SIC7            int             NULL,
    CategoryId8     int             NULL,
    SubCatId8       int             NULL,
    SIC8            int             NULL,
    CategoryId9     int             NULL,
    SubCatId9       int             NULL,
    SIC9            int             NULL,
    SIC10           int             NULL,
```

The relational table "CompanyAmbiguous" contains essentially the same columns as relational table "Company," as discussed in the previous subsection, as well as ten three-column triples, each comprising a numbered CategoryId column, a like-numbered SubCatId column, and a like-numbered SIC column. These triples contain business category and subcategory classifications and, where possible, a corresponding SIC. These columns contain newly extracted business category and subcategory information for incomplete or ambiguous records, as described in the previous subsection.

The SQL-like "CREATE TABLE" statement for the relational table "CompanyMerge," described and illustrated in the previous subsection, is provided below:

```
CREATE TABLE CompanyMerge
(
    CompanyId           binary(8)       NOT NULL  PRIMARY KEY NONCLUSTERED,
    NameSource          smallint        NULL,
    NameDate            smalldatetime   NULL,
    StreetAddSource     smallint        NULL,
    StreetAddDate       smalldatetime   NULL,
    MailAddSource       smallint        NULL,
    MailAddDate         smalldatetime   NULL,
    CitySource          smallint        NULL,
    CityDate            smalldatetime   NULL,
    StateSource         smallint        NULL,
    StateDate           smalldatetime   NULL,
    ZipSource           smallint        NULL,
    ZipDate             smalldatetime   NULL,
    EmailSource         smallint        NULL,
    EmailDate           smalldatetime   NULL,
    Web siteSource      smallint        NULL,
    Web siteDate        smalldatetime   NULL,
    Phone1Source        smallint        NULL,
    Phone1Date          smalldatetime   NULL,
    Phone2Source        smallint        NULL,
    Phone2Date          smalldatetime   NULL,
    TollFree1Source     smallint        NULL,
    TollFree1Date       smalldatetime   NULL,
    TollFree2Source     smallint        NULL,
    TollFree2Date       smalldatetime   NULL,
    FaxSource           smallint        NULL,
    FaxDate             smalldatetime   NULL,
    CtctSexSource       smallint        NULL,
    CtctSexDate         smalldatetime   NULL,
    CtctFirstSource     smallint        NULL,
    CtctFirstDate       smalldatetime   NULL,
    CtctMidSource       smallint        NULL,
    CtctMidDate         smalldatetime   NULL,
    CtctLastSource      smallint        NULL,
    CtctLastDate        smalldatetime   NULL,
    CtctPhoneSource     smallint        NULL,
    CtctPhoneDate       smalldatetime   NULL,
    LongitudeSource     smallint        NULL,
    LongitudeDate       smalldatetime   NULL,
    LatitudeSource      smallint        NULL,
    LatitudeDate        smalldatetime   NULL,
    SIC1                int             NULL,
    SIC2                int             NULL,
    SIC3                int             NULL,
    SIC5                int             NULL,
    SIC6                int             NULL
```

The relational table "CompanyMerge" contains data source and extraction date and time information for each of the fields, or columns, in relational tables "Company" and "CompanyExtra." As described in the previous subsection, the data source and data extraction time stamps are used in comparing newly extracted information to existing IMMM vendor database records in order to decide when existing IMMM vendor database records should be updated with the newly extracted information. In addition,

the relational table "CompanyMerge," contains a number of numbered SIC columns that contain SIC values that have been identified for a particular existing IMMM vendor database record but that have not been mapped to IMMM business categories and subcategories.

The SQL-like "CREATE TABLE" statement for instantiating the relational table "MergeCategory," described and illustrated in simplified form in the previous subsection, as well as an SQL-like "CREATE INDEX" statement that creates a clustered index on the relational table "MergeCategory," are provided below:

```
CREATE TABLE MergeCategory

    CompanyId       binary(8)       NOT NULL,
    CategoryId      int             NOT NULL,
    SubcategoryId   int             NULL,
    Added           bit             NOT NULL,
    Found           bit             NOT NULL,
    Source          smallint        NOT NULL,
    Date            smalldatetime   NOT NULL,
    SIC1            int             NULL,
    SIC2            int             NULL,
    SIC3            int             NULL,
    SIC4            int             NULL,
    SIC5            int             NULL,
    SIC6            int             NULL,
```

The relational table "MergeCategory" is used, as described in the previous subsection, during merging of newly extracted business category and subcategory information into the relational table "CategoryMap," Columns of the relational table "MergeCategory" not described and illustrated in the previous subsection include "Added," "Found," and six numbered SIC columns. The column "Added" contains bits, or Boolean values, that are reset prior to the start of the merge operation to indicate newly added entries resulting from the merge operation. The column "Found" is also reset upon the start of a merge operation, and indicates whether or not the entry has been updated during the merge operation. The numbered SIC columns may contain SIC values that have been mapped to the particular business category and subcategory.

SQL-like "CREATE TABLE" and "CREATE CLUSTERED INDEX" statements for the relational tables "NameIndex," "AddressIndex," "PhoneIndex," "Tempdb.dbo.MergeStatus," "tempdb.dbo.TokenMatch," and "tempdb.dbo.TokenSummary" are provided below:

```
CREATE TABLE NameIndex
(
    CompanyId       binary(8)       NOT NULL,
    NameToken       varchar(255)    NOT NULL,
    State           varchar(255)    NOT NULL,
    Ordinal         tinyint         NOT NULL,
)

CREATE CLUSTERED INDEX Name ON dbo.NameIndex
        (State, NameToken) WITH ALLOW_DUP_ROW

CREATE TABLE AddressIndex
(
    CompanyId       binary(8)       NOT NULL,
    AddressToken    varchar(255)    NOT NULL,
    State           varchar(255)    NOT NULL,
    Ordinal         tinyint         NOT NULL,
)

CREATE CLUSTERED INDEX Address ON dbo.AddressIndex
        (State, AddressToken) WITH ALLOW_DUP_ROW

CREATE TABLE PhoneIndex
(
    CompanyId       binary(8)       NOT NULL,
    Phone           varchar(255)    NOT NULL,
    State           varchar(255)    NOT NULL,
)

CREATE CLUSTERED INDEX Phone ON dbo.PhoneIndex
        (State, Phone) WITH ALLOW_DUP_ROW

CREATE TABLE tempdb.dbo.MergeStatus
(
    CompanyId       binary(8)   NOT NULL  PRIMARY KEY NONCLUSTERED,
    Added           bit         NOT NULL,
    Found           bit         NOT NULL,
)

CREATE TABLE tempdb.dbo.TokenMatch
(
    CompanyId       binary(8)       NOT NULL,
    Score           smallint        NOT NULL
)

CREATE TABLE tempdb.dbo.TokenSummary
(
    CompanyId       binary(8)       NOT NULL,
    TotalScore      smallint        NOT NULL
)
```

These tables are described and illustrated in the previous subsection, with the exception of table "tempdb.dbo.MergeStatus." The table "tempdb.dbo.MergeStatus" records volatile information about the state of the most recently conducted merge sequence. Each row in the table "tempdb.dbo.MergeStatus" represents a single company affected by the merge process, uniquely identified by the contents of the field "CompanyId." The field "Added" contains a Boolean value that indicates whether or not the company has been added to the master company database during the most recent merge process. The field "Found" contains a Boolean value that indicates whether or not the company

was identified, during the merge process, by entries in both an intermediate table and in the table "Company," and, therefore, whether or not a field-by-field data merge was performed on the corresponding entry in the table "Company."

Pseudocode and Flow Control Diagrams

Figure 7 is a flow control diagram for the continuous vendor database construction and update process of the present invention. This process comprises both manual steps executed by human technicians as well as automated steps implemented as software processes. This is a continuous process that may be run on a repeating, periodic basis or that may run more or less continuously during the life of a particular IMMM. In the first step 702, the next web site from which vendor information will be extracted is identified. In a preferred embodiment, this step is executed manually by an IMMM human technician. With increasingly sophisticated Internet knowbots and increasingly sophisticated artificial intelligence, this step may be automated. In addition, whether automated or executed manually, many different instantiations of this step and the other steps in Figure 7 may be conducted in parallel. Once the next web site is identified in step 702, a determination is made, in step 703, whether the information content represented by the web site is sufficiently large enough to justify generation of automated tools to extract the data. In a preferred embodiment, this determination is made by an IMMM technician, but, as with step 702, may also be made by an automated process. If there is not sufficient information content in the web site, as detected in step 703, information is extracted from the web site by an IMMM human technician in step 704 using browser-based extraction tools, and, in step 706, the extracted information is manually entered into the vendor database through manual entry tools that feature easy-to-use graphical interfaces. Following entry of the extracted vendor information into the vendor database, in step 706, the database creation and update process continues in a new iteration in step 702. If, on the other hand, it is determined that there is sufficient information within the web site to justify creation of a custom data extraction tool, then a custom data extraction tool is designed and built by an IMMM technician in step 708. The design of an actual custom data extraction tool is provided and discussed below. In step 710, the custom data extraction tool is executed, resulting in entry of vendor information extracted

from the web site into an intermediate table. If there are any hyperlinks, or URLs, in the web site pages examined, as detected in step 712, then a web crawler data extraction tool is run in step 714 to navigate the hyperlink graph emanating from each of the detected URLs in order to find and extract any vendor information from hyperlink web pages into an intermediate database. Determination of the presence of URLs in web site pages in step 712 may be carried out by an IMMM technician, or the URLs can be detected by an automated process which can then launch instances of the web crawler data extraction tool. In step 716, the newly extracted vendor information residing in one or more intermediate databases is merged into the vendor database using a merging tool. After merging of the newly extracted information, the process continues with a new iteration in step 702.

In the following, the automated data extraction tool for the American Society of Travel Agents web site is presented in pseudocode and discussed. This example will illustrate the process of building an automated data extraction tool. The first step in building a data extraction tool is to analyze the data content of the web site in order to develop a strategy for extracting the data. Next, an intermediate database for holding newly extracted information is developed. In the case of the American Society of Travel Agents web site, the following intermediate database is adequate for storing information gleaned from the American Society of Travel Agents web site for any particular travel agent:

Intermediatedatabase	
Name	varchar(255)
Address	varchar(255)
City	varchar(255)
State	varchar(255)
Zip	varchar(255)
Phone	varchar(255)
Fax	varchar(255)
Email	varchar(255)
Web site	varchar(255)
Contact	varchar(255)
Info	varchar(255)
Category	varchar(255)

An intermediate database is developed for a particular web site in order to directly store the particular types of vendor information available from that web site. In many cases, these types of vendor information will differ from the fields of the relational table "Company." The fields of the intermediate database can then later be used by a merging tool to construct the information required by the relational table "Company." Then, a simple HTML-based user interface may be constructed in order to allow an IMMM technician to choose various options, such as a database and/or relational table name into which newly extracted data will be placed, navigational parameters, and the business category for which information is to be extracted. The HTML-based user interface for the American Society of Travel Agents data extraction tool is shown below:

```
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" content="Microsoft FrontPage Express 2.0">
<title>Data Extraction Tool</title>
</head>

<body bgcolor="#FFFFFF">

<form method="POST">
    <p>Table: <input type="text" size="20" name="TableName"></p>
    <p><input type="checkbox" name="AllStates" value="ON">All
    States?</p>
    <p>Start At: <select name="StartState" size="1">
        <option>AL</option>
        <option>AK</option>
        <option>AZ</option>
        <option>AR</option>
        <option>CA</option>
        <option>CO</option>
        <option>DE</option>
        <option>... more ...</option>
    </select></p>
    <p>Category: <input type="text" size="20" name="Category"></p>
    <p><input type="submit" name="B1" value="Submit"></p>
</form>
</body>
</html>
```

This user interface includes a text entry field to specify the name of the relational table in which to place newly extracted data, a checkbox and selection list combination for specifying a subset of, or all, U.S. states from which to select travel agent information, and a text entry field for specifying the business category for the

extracted information, in this case most likely a business category related to travel agent services.

The next step in building an automated data extraction tool is to develop a navigation strategy for navigating the web site in order to extract information. In some web sites, the navigation may simply require navigation to a directory page, while for other, more elaborate web sites, the navigation strategy may include a process for filling in text entry boxes and check boxes on a search page within the web site and iteratively extracting the results of one or more searches. For even more elaborate web sites, the navigation strategy may include navigation through a hyperlink graph. Pseudocode for navigating the American Society of Travel Agents web site is shown below:

```
1  For each ZIP Code in the U.S.
2  // Call the AstaNet search engine.
3  // In this case we will use an HTTP GET command
4  Navigate to
5        "http://www.astanet.com/htbin/www_srch?SRCH_ZIP_AZ=" + current ZIP code
6           (plus additional parameters all set to default values)
7
8  Wait until navigation is complete
8
10    While TRUE
11    // AstaNet has a different format for the results page if there is
12    // one listing, than if there is more than one listing
13
14       Search the HTML for H2 tags
15
16          If an H2 tag is found    // just one listing
17             ParseSingleRecord
18             break
19       Else
20             ParseMultipleRecords
21
22          If there is not a "more" link on the page then
23             break
24
25       Navigate to the "more link
26
27       Wait until navigation is complete
```

The American Society of Travel Agents web site is, like the hypothetical web site used in the previous subsection and displayed in Figure 1, oriented towards zip codes. The navigation pseudocode, shown above, thus consists of an enclosing *for* loop in which each U.S. zip code is selected and used to solicit sublists of travel agents. On lines 4-5, the pseudocode routine navigates to a search page, supplying the selected zip code as an argument or parameter. On line 8, the pseudocode waits for the navigation to complete. Then, in the *while* loop beginning on line 10, the pseudocode parses the

results displayed on a search result page. First, on line 14, the pseudocode searches for header tags in the HTML representation of the search results page. If a header tag is found, on line 16, then the pseudocode parses the contents of the search results page as a single record on line 17. Otherwise, the pseudocode parses the search results page as containing multiple records, on line 20 and, if there are more pages in the search results page, navigates to the next of the pages on line 24 and waits for the navigation to complete on line 26.

The information in the search results page within the American Society of Travel Agents web site is formatted according to the following format:

Name
Address1
Address2 (optional)
City, State, ZIP
Phone
Fax (optional)
Email (optional)
Contact (optional)

With this format in mind, pseudocode for the "ParseSingleRecord" and "ParseMultipleRecords" pseudocode routines called from the above pseudocode routine are constructed as follows:

```
ParseSingleRecord
   Look for H2 tag
   H2 tag text is the Name

   Read the next line. That's Address1

   Read the next line

    If the line doesn't end with " #####"  or "#####-####" then
              This line is Address2
              Read the next line

    The last token in this line is ZIP
    The next to last token in this line is State
    The rest of this line is City

    Read the next line. This is the Phone
    Read the next line

    If this line contains "FAX: " then
           This line is the FAX
           Read the next line

    If this line contains an HREF ("A" tag) with a mailto type then
    Extract the Email from the HREF
    Read the next line

    If this line is non-empty then
              This line is the Contact
```

```
ParseMultipleRecords
   Find all TABLE tags in the HTML

   Ignore the first 3 and last two TABLEs, these do not contain company information

   For all other TABLE tags
      Scan all tags in the table
         For HREF ("A") tags
            If the link has a mailto type
               Extract the Email from the HREF
            Else
               Extract the Web site from the HREF

         For table cells ("TD", "TH") tags
            Extract the plain text for the cell

            Switch on the cell index
                                    First:
                                          The text is the Name
                                    Third:
                                          The first line is the Address
                                          The last token of the second line is the
                                          ZIP
                                          The next-to-last token is the State
                                          The remaining tokens are the City
               Fourth:
                  The first line that doesn't contain "fax" is the Phone
                  The first line that does contain "fax" is the FAX number
               Fifth:
                  The text is the Contact
               Sixth:
                  The text is the Info
```

Once the data has been extracted from the search results pages, the data extraction tool places the data into the intermediate database via SQL-like "SELECT" statements.

The web crawler data extraction tool has been described, by way of example, in the first subsection. This tool is used to obtain vendor data from web pages interlinked by a graph of hyperlinks. The web crawler data extraction tool is not tailored for particular types of web sites or to particular business categories or subcategories, but is instead a generalized tool that can be used to extract information from many different types of web sites. The web crawler data extraction tool is provided one or more URLs, or hyperlinks, that represent references to interconnected sets of web pages. The web crawler data extraction tool then navigates through the interconnected web pages and attempts to extract vendor information from each page. The web crawler data extraction tool can be designed to do depth first searching, in which numerous links are traversed before the information content of target web pages is analyzed, breadth-first searches, in which the information contained in a web page is analyzed prior to following any links that emanate from that page, or hybrid searches that vary the depth and breadth of the search depending on various factors,

including information contained on intermediate web pages. The web crawler data extraction tool can prioritize the order of link traversal based on heuristics. For example, the web crawler data extraction tool is designed to prefer traversals to pages associated with key words such as "about," "name," "location," and "company." The web crawler data extraction tool can enter marker records and navigational information into a database during navigation of a web site to facilitate future searching and to note the results of a search, including the inability to find vendor information by traversing particular hyperlinks.

The web crawler data extraction tool determines, based on certain initially analyzed criteria, whether or not a web page to which it has navigated may contain vendor information. For example, the web crawler data extraction tool may try to identify at least two types of information from the following informational categories within a web page: a full mailing address, an email address, and a phone or fax number. If two of the above categories of information are identified, the web crawler data extraction tool may search the entire page more thoroughly and extract any identified vendor information and save it to an intermediate table in the database. The web crawler data extraction tool may truncate searching along a particular navigational path or within a given web site once vendor information has been discovered on a web page along the navigational path or within the web site.

The web crawler data extraction tool may identify and extract the following types of information from web pages: (1) email addresses, identified as links that cause an email window to be activated when clicked; (2) mailing addresses, identified as lines of text that contain a state and zip code combination; (3) phone numbers, identified as a series of seven digits with a separating character, such as a hyphen, between the third and fourth digit; (4) fax numbers, identified as having the word "fax" prior to a phone number; and (5) any of the other data contained in columns of the relational table "Company" that can be identified based on formatting or text content criteria. Currently, data extracted by the web crawler data extraction tool is manually verified by an IMMM technician in order to evaluate the correctness

or suitability of the extracted data. However, automated verification routines may be developed in order to automate this verification process.

A high level, pseudocode algorithm for the merge tool is provided below:

```
1  Accept user information, such as data source, algorithm tuning parameters, etc.
2
3  If the master database index is out of date
4     Re-index the master database
5
6  For each record in source database
7     If record is a match with existing record in master database then
8        Do a field level merge from new record to existing record
9     Else if record is not match then
10       Add the new record to the master database
11    Else
12       Add the record to the ambiguous record table in master database
13
14 For each ambiguous record in the master database
15    Allow the user to determine if it is or is not a match with an existing record
16
17    If record is a match then
18       Do a field level merge from new record to existing record
19    Else if record is suitable
20       Add the new record to the master database
21
22 Print statistics, results
```

On line 1, the merge tool accepts certain parameters, such as an indication of the data source for data contained in the intermediate table being merged into the IMMM vendor database and various parameters that present user-level tuning of the merge algorithm. On line 3, the merge tool determines whether the master database index, represented by the relational tables "NameIndex," "AddressIndex," and "PhoneIndex," are out of date and, if so, rebuilds these tables in order to re-index the vendor database on line 4. In the *for* loop beginning on line 6, the merge tool considers each record stored in the intermediate table. If the record is determined to match with an existing IMMM vendor database entry, as determined on line 7, the merge tool does a field-by-field merge of the contents of the record in the intermediate table with the contents of the existing IMMM vendor database record on line 8. Otherwise, if the record in the intermediate database table is determined not to match any existing IMMM vendor database records, then a new record is created in the IMMM vendor database and the contents of the selected record from the intermediate database are copied into the newly created record on line 10. Otherwise, the selected record for the intermediate database is considered to be ambiguous, and is copied into

the relational table "CompanyAmbiguous." Next, in the *for* loop beginning on line 14, the contents of each record in the relational table "CompanyAmbiguous," are displayed through a graphical user interface on line 15 to an IMMM human technician. If the human technician determines that the contents of the record match an existing IMMM vendor database record, as determined on line 17, then IMMM technician merges the contents of the selected ambiguous record into the identified existing IMMM vendor database record on line 18. Otherwise, on line 19, if the human technician determines that the record contains sufficient and suitable data to create a new IMMM database record, then a new IMMM vendor database record is created on line 20 and the data contained in the ambiguous record is copied into the newly created record. Finally, on line 22, the merge tool may print out or present through a graphical user interface various statistics or results indicating the results of the merging operation.

In the first subsection, the matching algorithm employed by the merging tool was discussed and illustrated with examples. Again, the matching algorithm involves tokenizing the contents of certain data fields within a record of the intermediate database and then searching for matches within the IMMM vendor database indexing mechanism comprising the relational tables "NameIndex," "AddressIndex," and "PhoneIndex." The pseudocode and SQL-like statements based on the database schema described in the previous subsection are presented below to describe, in detail, various components of the matching algorithm.

The pseudocode for a high-level tokenizing routine is presented below:

```
Get Token
   Scan forward while the current char is not in (A-Za-z0-9#)

       // This is the start of the token

   If the current char is "#" then
      Return "#" as the token

   While more characters
               If current char is in (A-Za-z0-9) then
                  Add it to the token
               Else if current character is whitespace or "-" then
                  Break

   Convert the token to lowercase

       Return the token
```

This generalized tokenizing routine is repeatedly applied to the contents of a data field, such as a phone number containing data field or an address containing data field, in order to extract one or more all-lowercase tokens from the data field.

Data fields of entries in an intermediate database table are processed in a data field specific manner by one of the following three "sanitizing" routines that attempt to convert a tokenized name, tokenized address, or tokenized phone number into a common tokenized representation regardless of various peculiarities and differences in formatting of the original data fields. For example, the sanitizing routines will produce from a first phone-number-containing data field that contains a simple seven-digit phone number and a second phone-number-containing data field that contains the same seven-digit phone number along with an area code and a "1" in front of the are code, the same final sanitized token comprising the seven digits of the local exchange portion of the phone number without a formatting character between the third and fourth digits. Similarly, variations in the specification of the mailing address are removed by the address sanitizing routine in order to produce a finalized tokenization of the mailing address to which many different specifications of the mailing address converge through tokenizing and sanitizing processes.

Pseudocode for the routine "Sanitize Name" follows:

```
Sanitize Name

       For each token in the string
         If it is a common word ("the", "or", "a")
                        Remove it
      Else standardize common abbreviations
         // "corp." => "corporation"
         // "grp" => "group", etc.…
```

Name sanitization involves removing common words and expanding common abbreviations.

Pseudocode for the routine "Sanitize Address" follows:

```
Sanitize Address

   Find each token in the string

   If necessary fix it up:
      Convert to standardized postal abbreviations ("Street" => "St")
                        // see http://www.usps.gov/ncsc/lookups/abbr_suffix.txt
         Convert common abbreviations ("Intl" => "International")
         // also multi-token fix ups here, i.e. "Post Office" => "PO"
      Standardize directions ("North" => "N")
      Standardize numbers ("One" => "1")
```

```
Now look for the Street Suffix (i.e. "St", "Ave", "Blvd"...) and the Suite identifier
        (i.e. "Apt", "Rm", "Suite", ...)

If we found a suite identifier, then we combine together the suite identifier with
        the next following token, plus any additional following token of exactly
        one letter.
        // i.e. "Suite 110 A" becomes "suite_110a"

If we found a street suffix, then we can make the street name into a single token
        // i.e. "6205 N. Indian School Rd." will end up "6025 indian_school"

        First we find the street number. To do that we take the first token in the
                form #xxx after either the start of the string, or the suite
                substring, whichever comes first, but before the street suffix.

        Then we concatenate all tokens up to but not including the street suffix.
                We eat directionals during this process

        The street suffix we eat.

        If the street suffix may be a prefix (i.e. "Via", "Calle", etc.), we
                continue concatenating until we run out of tokens, we hit the
                suite substring, or we hit punctuation (line break, comma,
                semicolon, etc.)
        Else if it's a highway designation ("Jct", "Interstate", "Byp") and we
                haven't yet seen any non-directional strings, then we do append
                the street suffix, as well as one more token.
                // i.e. "405 N Black Valley Frwy" => 405 black_valley
                // but "11190 N Frwy 221" => "11190 Frwy_221"
        Else we will eat the next token if and only if it is a directional
                // i.e. "119 Katy Ave. N"

If we found a street and suite, reorder if necessary so the suite falls after the
street

Now make another pass to remove all remaining unattached street suffixes and
directions
```

This address sanitizing routine converts certain words to standardized postal abbreviations, converts common abbreviations to more standardized forms, coalesces common multi-token occurrences, and standardizes directions and numbers within postal addresses. The address sanitizing routine also coalesces street suffixes for street names into single tokens, with rewording in order to provide convergence.

The pseudocode for the routine "Sanitize Phone Number" follows:

```
Sanitize Phone Number
   Remove any leading "1"
   Extract just the digits
   Remove the area code
   Return the remaining digits as a single token
```

Phone number sanitization involves removing digits leading "1" that indicate long distance numbers, removing non-numeric characters, such as hyphens, and removing area codes.

The process of building the indexes within the relational tables "NameIndex," "AddressIndex," and "PhoneIndex" is described above in the first subsection, with example data. The pseudocode routine "Rebuild Index" that reindexes the IMMM vendor database is provided below:

```
Rebuild Index
        Empty the NameIndex, AddressIndex and PhoneIndex tables

        For each record in master database
          Sanitize the Name
          Tokenize the Name
          Add each Name token to the NameIndex table

          Sanitize the Address
          Tokenize the Address
          Add each Address token to the AddressIndex table

          For each phone number in (Phone1, Phone2, TollFree1, TollFree2)
             Sanitize the phone number
               Add the result to the PhoneIndex table
```

The merging tool, described above in the present subsection and in the first subsection, employs a matching algorithm to identify vendors currently described in the IMMM vendor database that may correspond to newly extracted vendor information residing in the record of an intermediate database table. Pseudocode for the routine "Find Match" employed by the merging tool is provided below:

```
1  Find Match
2     Set the threshold to default
3     If Name or Address not present, Record is ambiguous
4     Else
5        Sanitize the Name
6        Tokenize the Name
7        For each Name token call squid_MatchNameToken
8           This will add points for each company that has the given name token
9           // (in all cases the points are parameterized)
10
11          If this is the first name token in the input record then
12             It will add more points for each company that also has this for its
13                first name token
14          Else
15             It will add more points for any company that has this token, and also
16                has the same previous name token as the input record
17
18       For each name token after the second, add points to the threshold
19
20       Sanitize the Address
21       Tokenize the Address
22       For each Address token call squid_MatchAddressToken
23          This will add points for each company that has the given address token
24
25          If this is the first address token in the input record then
26             It will add more points for each company that also has this for its
27                first address token
28          Else
29             It will add more points for any company that has this token, and also
30                has the same previous address token as the input record
31
32       For each address token after the second, add points to the threshold
33
34       For each phone number in (Phone1, Phone2, TollFree1, TollFree2)
35          Sanitize the phone number
36          Call squid_MatchPhoneToken
37             This will add points for each company with a matching phone number
38
39       If the input record has a city
40          Call squid_MatchCityToken
41             This will add points for each company with an exact city match
42             (ignoring punctuation and case insensitive)
43
44          Add points to the threshold
45
46       Call squid_GetScores
47
48       This will return the highest match scores
49
50       If the highest score is within a user specified range of the threshold
51          Record is ambiguous
52       Else if it is greater or equal to the threshold
53          Record is a match
54       Else
55          Record is not a match
```

The routine Find Match, as described above in the first section, identifies potentially matches to already described vendors by placing entries into the relational table "tempdb.dbo.TokenMatch," described above. Then, all entries in the relational table "tempdb.dboTokenMatch" for a particular vendor identified by a CompanyId are summed and coalesced into a single entry in the relational table

"tempdb.dbo.TokenSummary." Finally, one or more CompanyIds are selected from entries in the relational table "tempdb.dbo.TokenSummary" having the highest total score. The selected CompanyIds correspond to vendors already described in the IMMM vendor database that is most likely described by the intermediate table entry that is being matched to the indexing mechanism.

On line 2, Find Match sets two threshold values to default values. The threshold values will be used, at the end of the routine "FindMatch," in order to determine whether the intermediate database table record being matched via the indexing process matches an existing IMMM vendor database entry, is ambiguous, or represents a new, previously undescribed vendor. A total score having a value lower than the lower threshold value indicates that the intermediate database table record represents a new, previously undescribed vendor. A total score falling between the lower threshold value and an upper threshold value indicates that the intermediate database table record is ambiguous, neither clearly representing a new, previously undescribed vendor nor clearly and unambiguously matching an existing entry within the IMMM vendor database. A total score equal to or exceeding the upper threshold values indicates that the intermediate database table record unambiguously corresponds to an existing IMMM vendor database entry. The lower and upper threshold values are modified, during execution of the routine "Find Match," in order to normalize the threshold values with respect to the number of tokens generated from the name, address, and phone number fields that are matched via the indexing process to the entries in the indexing tables.

On lines 5 and 6, Find Match sanitizes and tokenizes the contents of the name field in the intermediate database record and then, on line 7, calls the routine "squid_MatchNameToken" to generate entries in the relational table "tempdb.dbo.TokenMatch" that represent matches between the generated tokens and entries in the indexing tables "NameIndex," "AddressIndex", and "PhoneIndex." Thresholds are adjusted, on line 18, to normalize the thresholds with respect to numbers of generated tokens greater than two.

A similar indexing process is conducted by the routine "Find Match" on lines 20-32, for address fields within the intermediate database table record, and, on lines 33-36, for the phone number fields within the intermediate database table record. The routine "squid_MatchAddressToken" is called to generate entries in the relational table "tempdb.dbo.TokenMatch" for tokens generated from address fields, and the routine "squid_MatchPhoneToken" is called to generate entries in the relational table "tempdb.dbo.TokenMatch" corresponding to tokens generated from phone number fields of the intermediate database table record. If the intermediate database table record includes a city field, then a similar indexing process is carried out on lines 38-43 by the routine "FindMatch" via a call to the routine "squid_MatchCityToken." On line 45, "FindMatch" calls the routine "squid_GetScores" to find the highest total score or total scores for existing IMMM vendor database entries, or, in other words, to find previously described vendors that most closely correspond to the data contained within the intermediate table database record. If the highest score is equal to or greater than the lower threshold value, but less than the upper threshold value, as detected by the routine "FindMatch" on line 49, then the routine "FindMatch" determines that the record is ambiguous on line 50. Otherwise, if the highest total score is greater than or equal to the upper threshold value, as detected by the routine "FindMatch" on line 51, then the routine "FindMatch" determines that the intermediate database table record corresponds to an existing, previously described vendor within the IMMM vendor database on line 52. Otherwise, the routine "FindMatch" determines, on line 54, that the vendor information within the intermediate database table record corresponds to a new, previously undescribed vendor.

SQL-like procedures for the routines called from the above-described routine "FindMatch" are provided below:

```
CREATE PROCEDURE squid_MatchNameToken
   @sState varchar(255),
   @sToken varchar(255),
   @sPrevToken varchar(255),
   @wScore smallint,
   @wScoreFirst smallint,
   @wScorePrev smallint
AS
   INSERT INTO tempdb.dbo.TokenMatch
      SELECT CompanyId, @wScore FROM NameIndex
         WHERE State = @sState AND NameToken = @sToken
```

```
    IF @sPrevToken = ''
       INSERT INTO tempdb.dbo.TokenMatch
          SELECT CompanyId, @wScoreFirst FROM NameIndex
             WHERE State = @sState AND NameToken = @sToken AND Ordinal = 0
    ELSE
       INSERT INTO tempdb.dbo.TokenMatch
          SELECT CompanyId, @wScorePrev FROM NameIndex
                              WHERE State = @sState AND NameToken = @sPrevToken AND
                                    CompanyId IN
                                          (SELECT CompanyId FROM NameIndex WHERE
                                                State = @sState AND NameToken = @sToken)
                                    AND (Ordinal + 1) IN
                                          (SELECT Ordinal FROM NameIndex WHERE
                                                State = @sState AND NameToken = @sToken)


CREATE PROCEDURE squid_MatchAddressToken
   @sState varchar(255),
   @sToken varchar(255),
   @sPrevToken varchar(255),
   @wScore smallint,
   @wScoreFirst smallint,
   @wScorePrev smallint
AS
   INSERT INTO tempdb.dbo.TokenMatch
      SELECT CompanyId, @wScore FROM AddressIndex
         WHERE State = @sState AND AddressToken = @sToken

   IF @sPrevToken = ''
      INSERT INTO tempdb.dbo.TokenMatch
         SELECT CompanyId, @wScoreFirst FROM AddressIndex
            WHERE State = @sState AND AddressToken = @sToken AND Ordinal = 0
   ELSE
      INSERT INTO tempdb.dbo.TokenMatch
         SELECT CompanyId, @wScorePrev FROM AddressIndex
            WHERE State = @sState AND AddressToken = @sPrevToken AND
                                    CompanyId IN
                                          (SELECT CompanyId FROM AddressIndex WHERE
                                                State = @sState AND AddressToken =
                                          @sToken)
                                    AND (Ordinal + 1) IN
                                          (SELECT Ordinal FROM AddressIndex WHERE
                                                State = @sState AND AddressToken =
                                          @sToken)


CREATE PROCEDURE squid_MatchPhoneToken
   @sState varchar(255),
   @sToken varchar(255),
   @wScore smallint
AS
   INSERT INTO tempdb.dbo.TokenMatch
      SELECT CompanyId, @wScore FROM PhoneIndex
         WHERE State = @sState AND Phone = @sToken

CREATE PROCEDURE squid_MatchCityToken
   @sState varchar(255),
   @sCity varchar(255),
   @wScore smallint
AS
   INSERT INTO tempdb.dbo.TokenMatch
      SELECT CompanyId, @wScore FROM Company
         WHERE State = @sState AND City = @sCity


CREATE PROCEDURE squid_GetScores
AS
   SELECT CompanyId, TotalScore = SUM(Score)
               FROM tempdb.dbo.TokenMatch
                     GROUP BY CompanyId
                           ORDER BY TotalScore DESC
```

In all of these SQL-like procedures, a token to be matched against entries in the indexing tables, as well as, in certain cases, a previous token that occurred within the analyzed field of the intermediate database record prior to the current token, is passed into the procedure as a parameter, specified by a character string starting with the character "@." Also, a score is passed into the procedures to be entered as the score in entries stored in the relational table "tempdb.dbo.TokenMatch." These procedures consist of SQL-like statements based on the data schema described above in the second subsection.

The merging tool, having identified an intermediate database table record as corresponding to an existing IMMM vendor database entry, then applies the following algorithm, on a field-by-field basis, to each field of intermediate database table record and the corresponding field of the existing IMMM vendor database entry:

```
If the input record has data
        If the master record has no data, or the input source has a higher trust
           level, or the input source has an equal trust level and a more recent
           timestamp then

        If the input record has different data than the master record then
           Update the master record data in the Company table
           Update the appropriate index table if necessary

        Update the merge table to contain the input source and date
```

Thus, any newly extracted data residing in the intermediate database table record that is not already contained in an existing IMMM vendor database record, that is more reliable than different corresponding data present in an existing IMMM vendor database record, or is equally reliable but more recent than corresponding data in an existing IMMM vendor database record, is copied from the intermediate database table record into the appropriate field of the existing IMMM vendor database record. When data is copied from the intermediate database table record into the existing IMMM vendor database record, the IMMM vendor database tables that contain data source and data time stamp values associated with the copied fields are updated and the indexing tables are updated.

When an IMMM technician processes an ambiguous record, it is useful for the IMMM technician to refer to the CompanyId, company data, and match scores

for various companies that may be referred to by the contents of the ambiguous record. This information can be obtained via the following SQL-like routines:

```
CREATE PROCEDURE squid_PrepareMatches
AS
        DELETE FROM  tempdb.dbo.TokenSummary

        INSERT INTO tempdb.dbo.TokenSummary
          SELECT CompanyId, TotalScore = SUM(Score)
            FROM tempdb.dbo.TokenMatch
              GROUP BY CompanyId

CREATE PROCEDURE squid_GetMatches
AS
        SELECT Company.CompanyId, Name, StreetAddress, City, State, Phone1, Phone2,
TollFree1, TotalScore
          FROM Company INNER JOIN tempdb.dbo.TokenSummary
            ON Company.CompanyId =
tempdb.dbo.TokenSummary.CompanyId
              ORDER BY TotalScore DESC
```

Although the present invention has been described in terms of a particular embodiment, it is not intended that the invention should be limited to this embodiment. Modifications within the spirit of the invention will be apparent to those skilled in the art. For example, the IMMM vendor database may be implemented as an ad hoc database, using common programming languages and operating system interfaces, or may be implemented using any of the various types of database management systems, including hierarchical, networking, relational, object-oriented, and hybrid database management systems. Many different techniques may be employed to determine that newly acquired information matches an existing record, and many different methods can be employed to recognize and parse vendor information from web pages. There are a large number of database schemas that may be designed according to the teachings of the present invention to include the required vendor data and organized to facilitate ongoing database creation, maintenance, and update, as discussed above. Alternate database schemas may provide for, and accommodate for, storage of new types of vendor information, or vendor information formatted and represented in different ways. The steps in the database creation and update processes described above may be somewhat varied in sequence and may be more or less automated than indicated in the above description. The balance point

between human intervention and full atomization is determined by the competing goals of obtaining and storing as comprehensive and reliable data as possible but doing so in the most cost-effective and time-efficient manner as possible. If there are improvements in automated tools for hyperlink graph navigation and web page information processing, the steps will necessarily evolve towards greater and more complete automation and less need for human intervention during database creation and updating processes.

The foregoing description, for purposes of explanation, used specific nomenclature to provide a thorough understanding of the invention. However, it will be apparent to one skilled in the art that the specific details are not required in order to practice the invention. In other instances, well-known components are shown in block diagram form in order to avoid unnecessary distraction from the underlying invention. Thus, the foregoing descriptions of specific embodiments of the present invention are presented for purposes of illustration and description; they are not intended to be exhaustive or to limit the invention to the precise forms disclosed, obviously many modifications and variations are possible in view of the above teachings. The embodiments were chosen and described in order to best explain the principles of the invention and its practical applications and to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalents:

Claims

1. A database, stored in a computer-readable format on a memory device, containing vendor data for a number of previously identified vendors, the database organized for frequent addition of vendor data for newly identified vendors and frequent updating of vendor data for previously identified vendors, the database comprising:

a vendor data component that stores basic vendor data for previously identified vendors, including previously identified vendors' names, addresses, phone numbers, zip codes, email addresses, and fax numbers;

a category mapping component that represents a many-to-many relationship between previously identified vendors and categories of products and services that can be offered by vendors;

a vendor data merge information component that stores an indication of a data source and timestamp for basic vendor data contained in the vendor data component; and

an indexing component that indexes the basic vendor data to allow for rapid identification of previously identified vendors that may be described by newly obtained vendor information.

2. The database of claim 1 implemented within a relational database management system as a collection of relational tables and procedures.

3. The database of claim 2 wherein the vendor data component that stores basic vendor data for previously identified vendors further stores:

vendors' street addresses and a mailing addresses;

vendors' web site addresses;

vendors' secondary phone numbers; and

vendors' toll-free phone numbers.

4. The database of claim 3 wherein the vendor data component that stores basic vendor data for previously identified vendors further stores longitudes and latitudes for vendors.

5. The database of claim 3 wherein the vendor data component comprises a number of relational database tables.

6. The database of claim 2 wherein the category mapping component that represents a many-to-many relationship between previously identified vendors and categories of products and services that can be offered by vendors is implemented as a relational database table that includes columns containing:

identifiers of companies;

identifiers of major business categories for which the vendor offers an item for sale; and

identifiers of business subcategories for which the vendor offers an item for sale.

7. The database of claim 2 wherein the vendor data merge information component that stores an indication of a data source and timestamp for basic vendor data contained in the vendor data component is implemented as a number of relational database tables containing data source and timestamp columns for data columns within the number of relational tables that together compose the vendor data component of the database.

8. The database of claim 2 wherein the indexing component that indexes the basic vendor data comprises a number of relational database tables, each having a column containing identifiers that identify vendors and columns that contain tokens.

9. The database of claim 8 wherein a token represents the results of a tokenizing process that transforms a particular unit of vendor information stored in the vendor data component and that may be specified in a number of different forms into a common token that results from performing the tokenizing process on a number of the different forms that specify the particular unit of vendor information.

10. The database of claim 9 wherein the indexing component includes the following token tables:

an address token table that stores tokenized addresses;

a name token table that stores tokenized names; and

a phone number token table that stores tokenized phone numbers.

11. A method for periodically enhancing a vendor database by extracting vendor information from a digital communications medium, the method comprising:

identifying an information site within the digital communications medium that may contain vendor data for a number of vendors;

determining whether the identified information site contains a desirable quantity of vendor information;

when the identified information site contains a desirable quantity of vendor information,

extracting vendor information from the information site by an automated process, and

placing the extracted information into a temporary storage component; and

merging the extracted vendor information stored in the temporary storage component into the vendor database.

12. The method of claim 11 wherein the digital communications medium is the Internet and wherein the information site is a web page.

13. The method of claim 12 wherein identifying an information site within the digital communications medium that may contain vendor data for a number of vendors further includes analyzing the informational content of the web page, and related web pages, by a human technician.

14. The method of claim 12 wherein identifying an information site within the digital communications medium that may contain vendor data for a number of vendors further includes analyzing the informational content of the web page, and

related web pages, by an automated process implemented as a computer program that parses information from the computer-readable representation of the web page and related web pages.

15. The method of claim 12 wherein determining whether the identified information site contains a desirable quantity of vendor information comprises determining whether the number of vendors for which information can be extracted from the web page, and related web pages, is greater than a threshold value.

16. The method of claim 12 further including, when the identified information site contains a desirable quantity of vendor information, first implementing a computer program to parse vendor information from the computer-readable representation of the web page and related web pages.

17. The method of claim 12 further including, when the identified information site contains vendor information about fewer than a threshold number of vendors,

manually extracting vendor data from the identified information site; and manually merging the manually extracted vendor data into the vendor database.

18. The method of claim 12 wherein, when the information site contains references to additional information sites,
launching an automated navigational data extraction tool to automatically navigate the referenced additional information sites to extract vendor information from the referenced additional information sites and placing the extracted information into the temporary storage component.

19. The method of claim 11 wherein merging the extracted vendor information stored in the temporary storage component into the vendor database further includes:

for each record of vendor information stored in the temporary storage component,

tokenizing a number of fields within the record;

matching the tokenized fields to tokenized vendor data in an indexing component of the vendor database to generate intermediate matches;

summing the generated intermediate matches for each vendor described by data stored in the vendor database in order to produce a score for each vendor described by data stored in the vendor database for which matches are generated;

determining whether the highest produced score represents a definite match, and

when the highest produced score represents a definite match, merging fields within the record into corresponding fields within a record in the vendor database that describes a vendor associated with the highest score.

20. The method of claim 19 wherein determining whether the highest produced score represents a definite match further includes:

determining whether the highest score exceeds an upper threshold, whether the highest score falls between the upper and a lower threshold, or whether the highest score falls below the lower threshold.

21. The method of claim 20 wherein, when the highest score exceeds an upper threshold, the highest score represents a definite match, wherein, when the highest score falls between the upper and a lower threshold, the highest score represents an ambiguity, and wherein, when the highest score falls below the lower threshold, the highest score indicates that the record describes a vendor not described by a record in the vendor database.

22. The method of claim 21 wherein records for which the highest score represents an ambiguity are placed in an ambiguous records storage component for later analysis.

23. The method of claim 21 wherein records for which the highest score indicates that the record describes a vendor not described by a record in the vendor database are used to create new records within the vendor database.

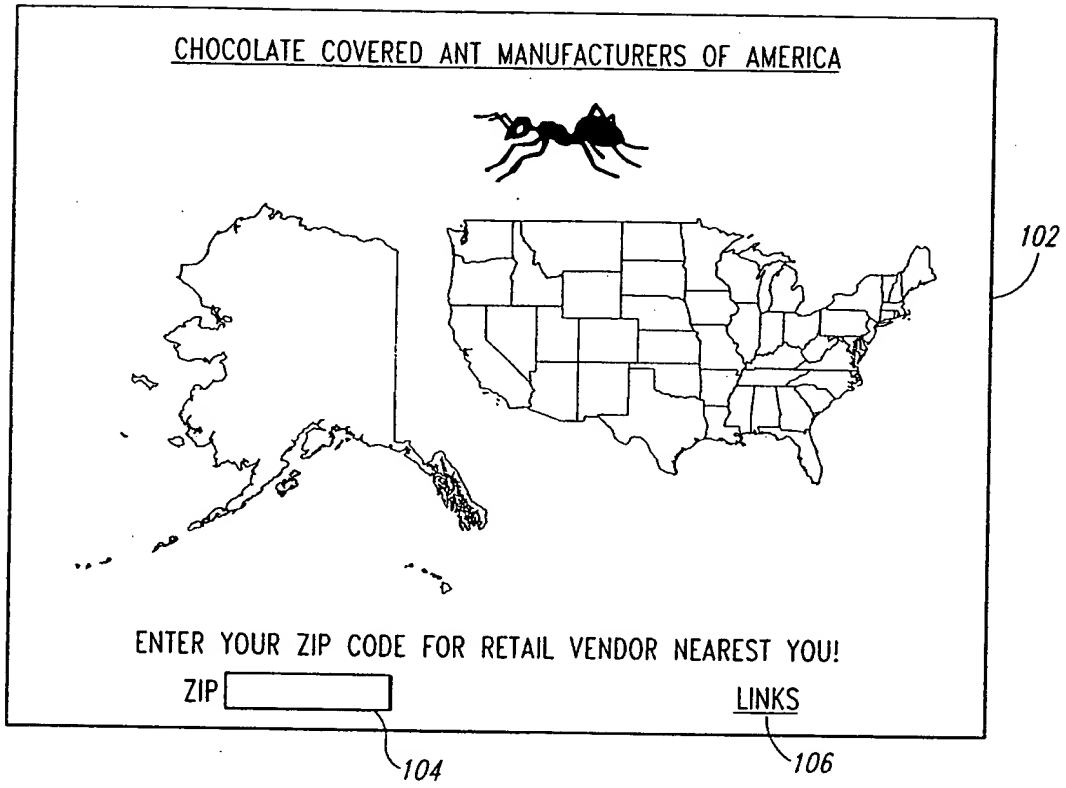

CHOCOLATE COVERED ANT MANUFACTURERS OF AMERICA
102
ENTER YOUR ZIP CODE FOR RETAIL VENDOR NEAREST YOU!
ZIP
LINKS
104
106

Fig. 1

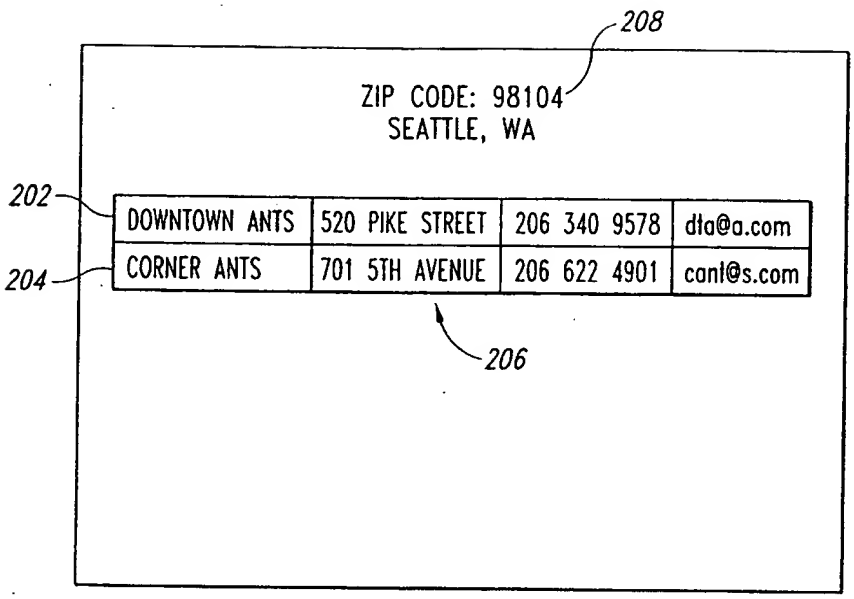

208
ZIP CODE: 98104
SEATTLE, WA
202
204
DOWNTOWN ANTS
520 PIKE STREET
206 340 9578
dta@a.com
CORNER ANTS
701 5TH AVENUE
206 622 4901
cant@s.com
206

Fig. 2

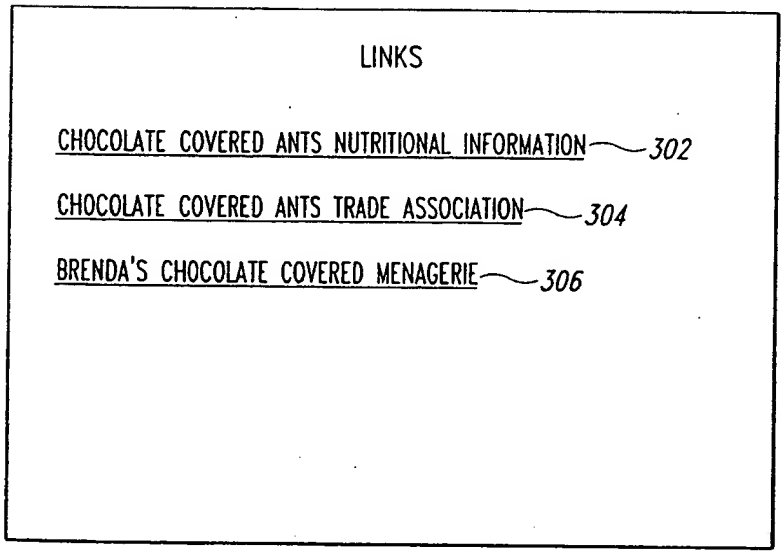

LINKS
CHOCOLATE COVERED ANTS NUTRITIONAL INFORMATION
302
CHOCOLATE COVERED ANTS TRADE ASSOCIATION
304
BRENDA'S CHOCOLATE COVERED MENAGERIE
306

Fig. 3

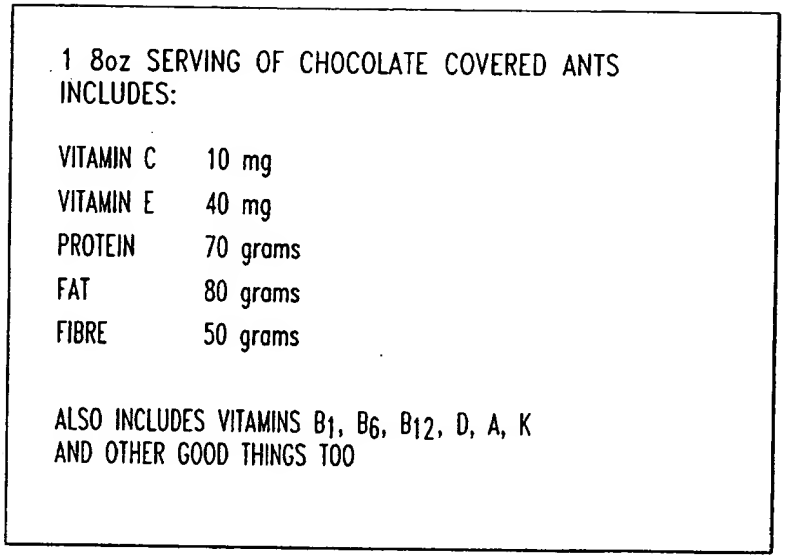

1 8oz SERVING OF CHOCOLATE COVERED ANTS INCLUDES:
VITAMIN C 10 mg
VITAMIN E 40 mg
PROTEIN 70 grams
FAT 80 grams
FIBRE 50 grams
ALSO INCLUDES VITAMINS B1, B6, B12, D, A, K AND OTHER GOOD THINGS TOO

Fig. 4

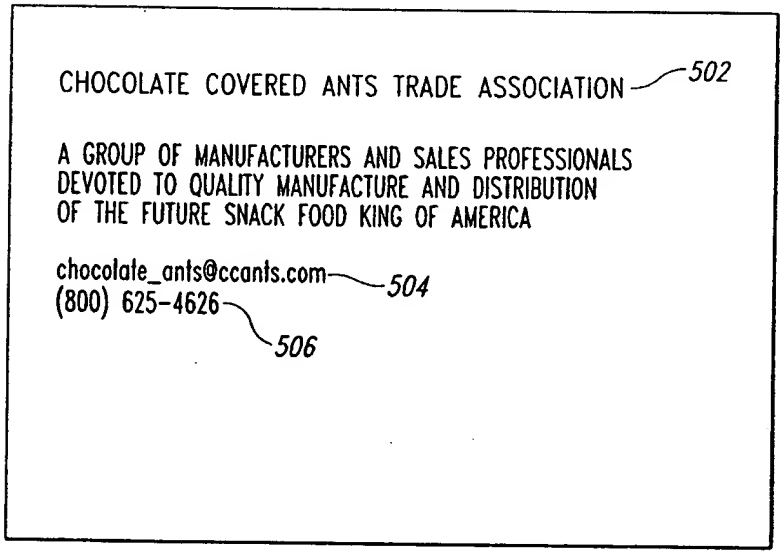

CHOCOLATE COVERED ANTS TRADE ASSOCIATION
502
A GROUP OF MANUFACTURERS AND SALES PROFESSIONALS DEVOTED TO QUALITY MANUFACTURE AND DISTRIBUTION OF THE FUTURE SNACK FOOD KING OF AMERICA
chocolate_ants@ccants.com
504
(800) 625-4626
506

Fig. 5

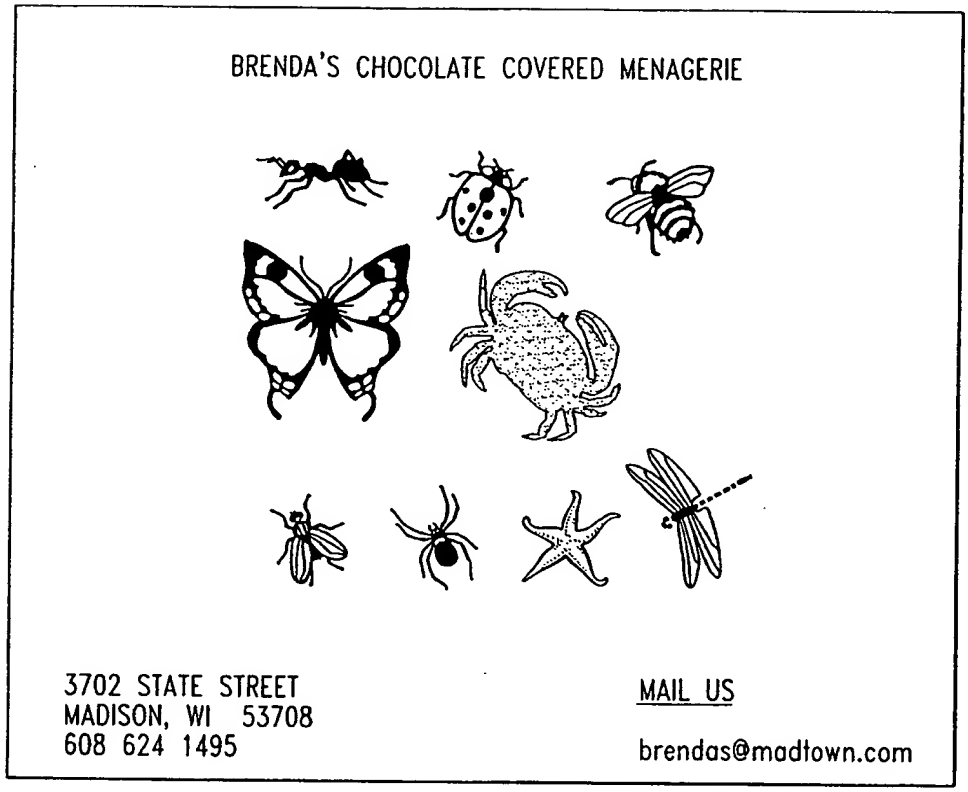

BRENDA'S CHOCOLATE COVERED MENAGERIE
3702 STATE STREET
MADISON, WI 53708
608 624 1495
MAIL US
brendas@madtown.com

Fig. 6

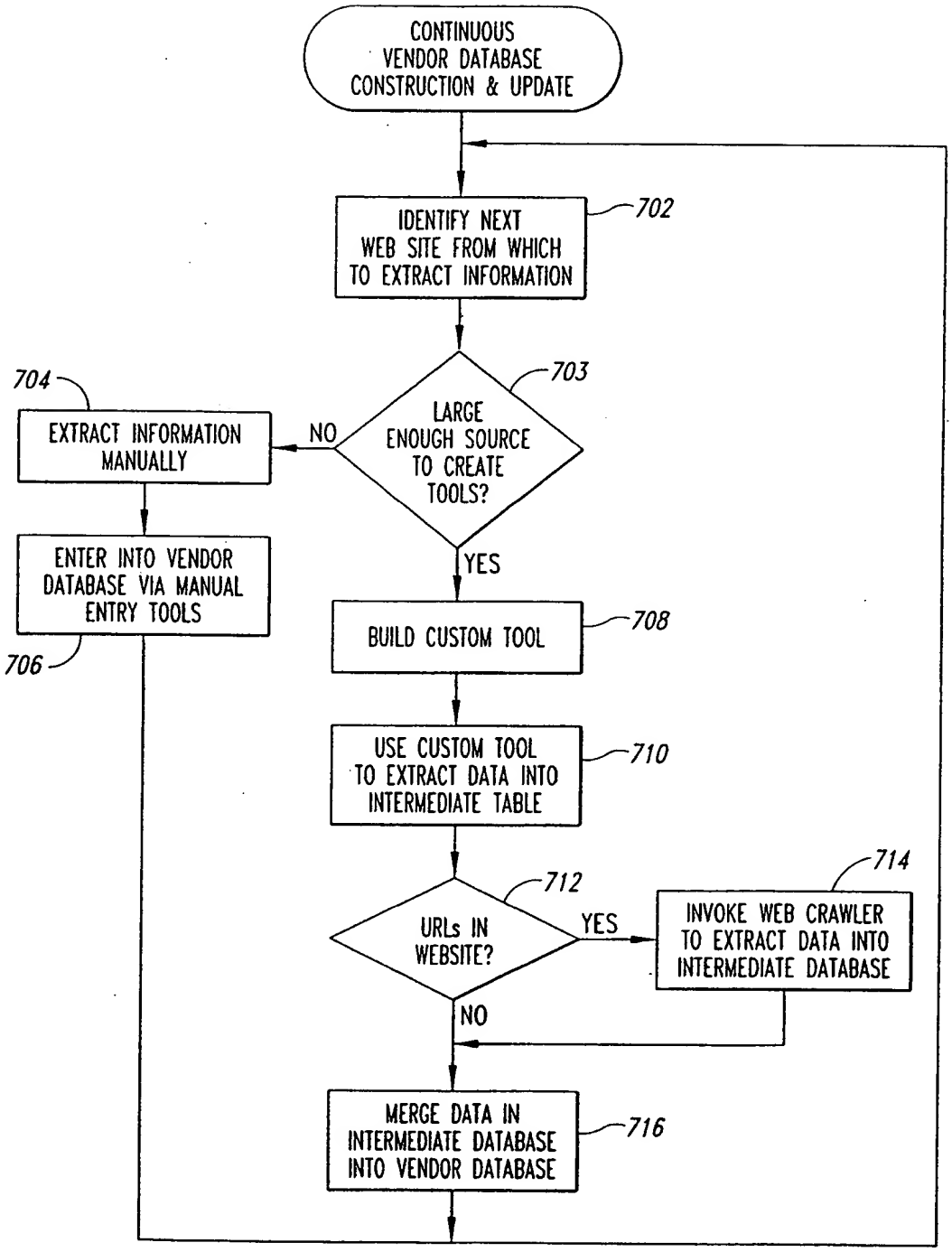

CONTINUOUS VENDOR DATABASE CONSTRUCTION & UPDATE
IDENTIFY NEXT WEB SITE FROM WHICH TO EXTRACT INFORMATION
702
LARGE ENOUGH SOURCE TO CREATE TOOLS?
703
NO
YES
EXTRACT INFORMATION MANUALLY
704
ENTER INTO VENDOR DATABASE VIA MANUAL ENTRY TOOLS
706
BUILD CUSTOM TOOL
708
USE CUSTOM TOOL TO EXTRACT DATA INTO INTERMEDIATE TABLE
710
URLs IN WEBSITE?
712
YES
NO
INVOKE WEB CRAWLER TO EXTRACT DATA INTO INTERMEDIATE DATABASE
714
MERGE DATA IN INTERMEDIATE DATABASE INTO VENDOR DATABASE
716

Fig. 7